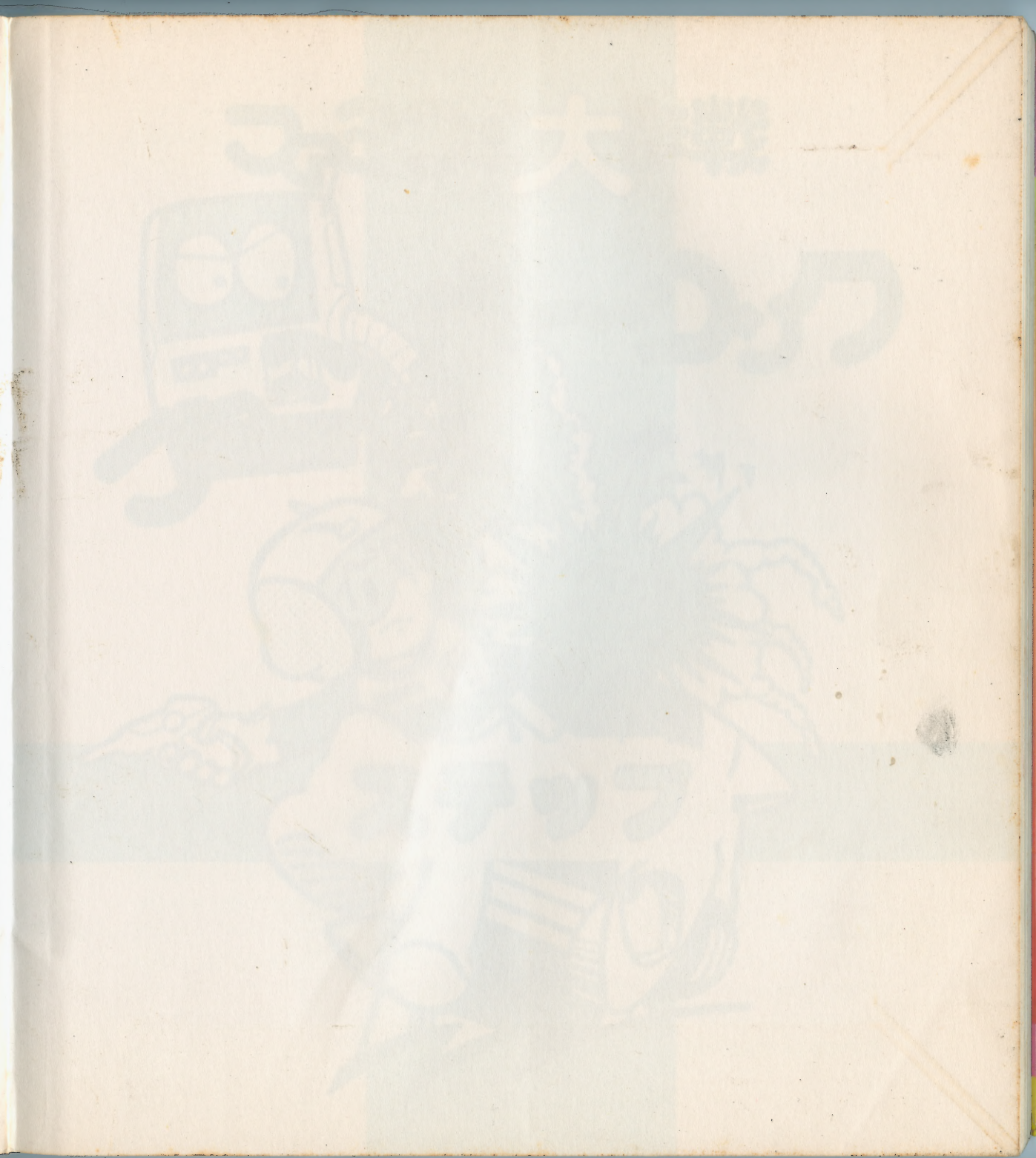
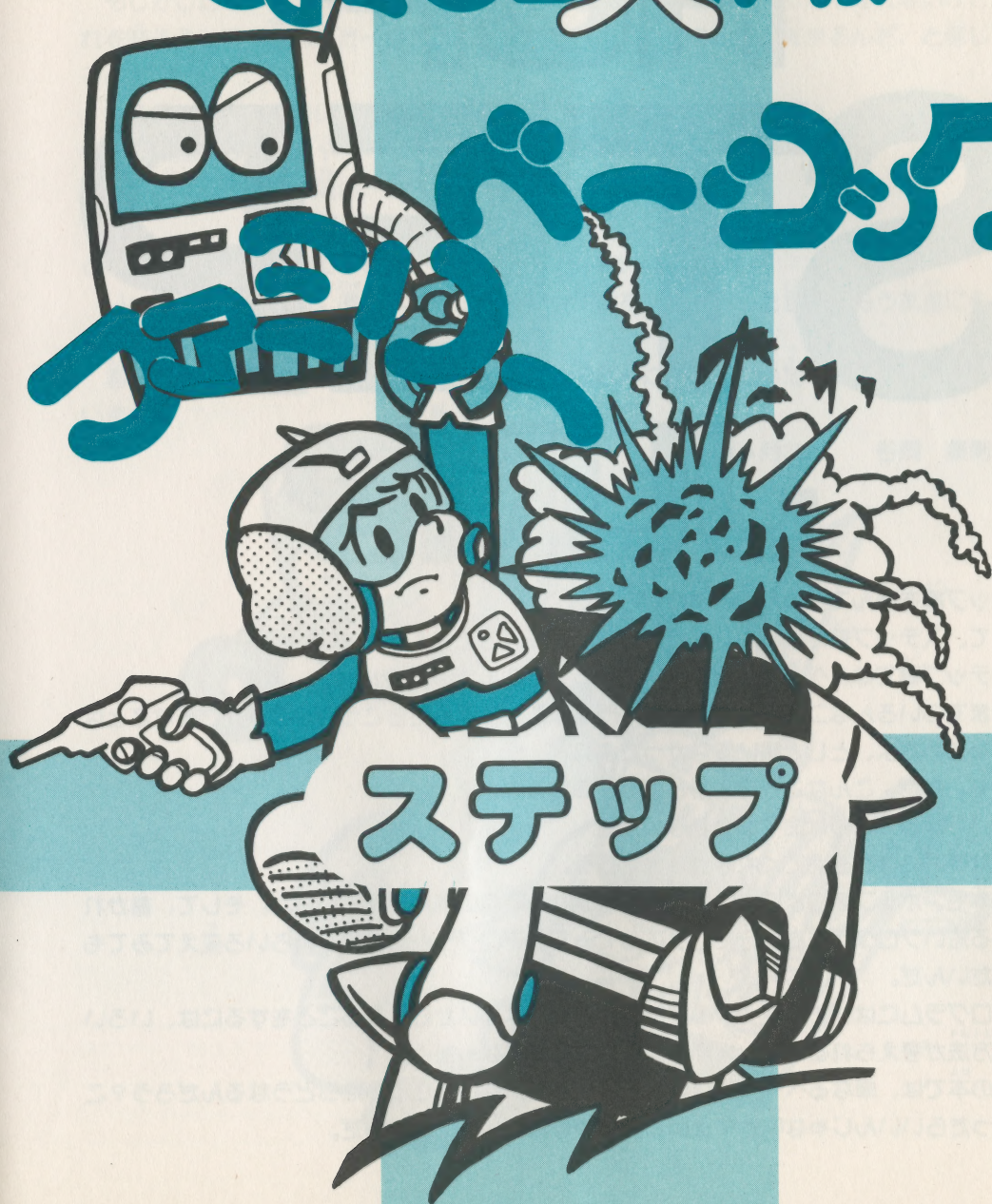
ファミコンを
どんどん使いこなせ

ファミコン大作戦

ファミリーベーシック™ ステップ

地球防衛軍 編著
司令　谷岡康則

誠文堂新光社

ファミコン大作戦
ファミリーベーシック
ステップ

HOP STEP JUMP

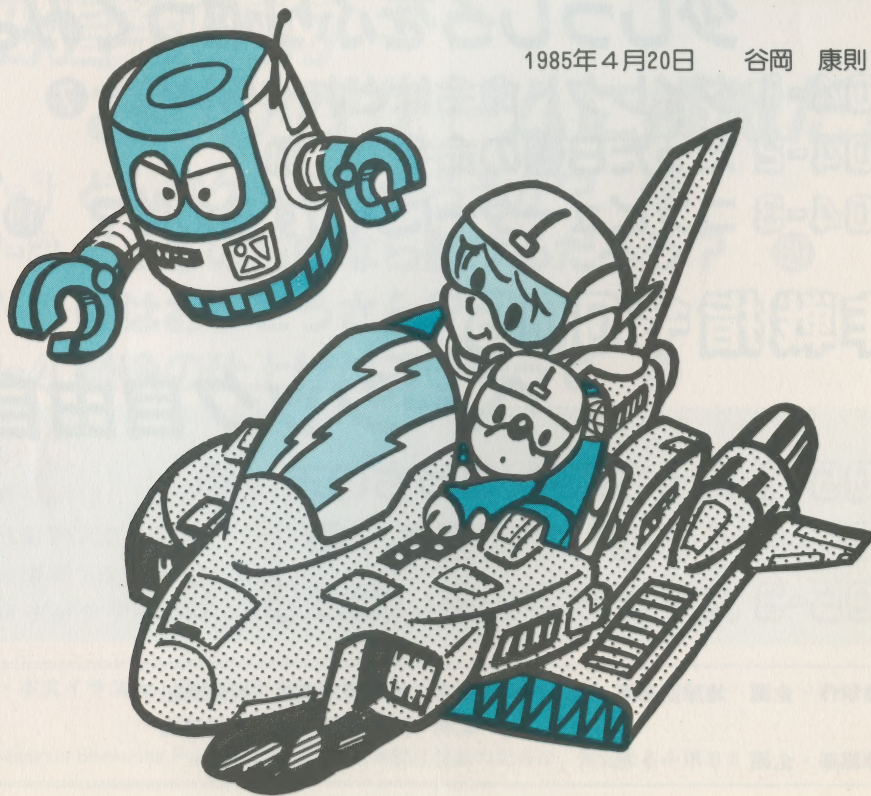
ホップ編を読んでくれた君、ありがとう。
そして、ステップ編から参加してくれる君、がんばろうね！
ステップ編では、ベーシックについて少しつっこんでみよう。
今までのいろんなコンピューターの入門書って、こうしたらこうなります、こんなことはしちゃだめだよ、としか書いてなかったよね。
でも、なぜ、こんなことをするとエラーになるのか？
なぜ、こんな命令が必要なのか？
こんな疑問には、答えてくれなかっただろう。
この本も、ホップ編のように、最初のページから順番に読んでほしいんだ。そして、書かれてある短いプログラムを自分で打ち込んでみて、プログラムの数値をいろいろ変えてみてもらいたいんだ。
プログラムには､「たった一つの正解」というのはないから、あることをするには、いろいろな方法が考えられるんだ。大事なのはその考え方なんだよ。
この本では、単なるベーシックの解説じゃなく、ここをこうしたらどうなるんだろう？こうやったらいいんじゃないか？といろんなやり方を考えてみたんだ。

ファミコンだけでなく、他のパソコンはどうなんだろう？というような疑問にも、ふれているんだよ。
もしかしたら、ファミコンだけで遊ぶには、必要のないこともあるかも知れないけど、それを知ることで、君はまた一歩、コンピューターについての理解が深まるんだ、と信じているよ。
教えられたこと、書いてあることなら知ってるけど、あとは知らない、それじゃあ、つまらないと思わないかい？
コンピューターと人間の一番大きな違いは、

「なぜ？」

と疑問を持てるか、持てないか、ということではないのかな？
いっしょに考えよう。考えながらこのステップ編をマスターした君は、もう友達にも教えることができるようになれるはずだぞ！
君も、そして、君のお父さん、お母さん、お兄さんにも一緒に参加してほしい、と思っています。

1985年４月20日　　谷岡　康則

もくじ

作戦指令004
少しうしろをふり返ってみよう

004-1 ダイレクト命令はどれぐらい？ ⑦
004-2 たった8個の命令で ⑭
004-3 コンピューターに入力するのは？ ⑲

作戦指令005
ベーシック自由自在

005-1 あっちの道、こっちの道 ㉙
005-2 くり返して何回も ㉞
005-3 文字と数字のあいだがら ㊽

●制作・企画　地球防衛軍　谷岡康則・岩田　裕・鈴木康夫・真藤美紀子・
簑藤　桂・北村　順・初瀬　龍・美崎靖之
●編集・企画　リポート社

作戦指令006
テレビへ落書き、3つの方法

006-1 PRINT文にも、いろいろあるぞ！ 56
006-2 ファミコンは何重人格？ 61
006-3 スプライトが重なると。 67
006-4 スプライトを動かすには？ 73
006-5 MOVEというのは、動くという意味 77

作戦指令007
おっかけゲームに挑戦だ！

007-1 どんなゲームを作ろうか？ 86
007-2 どっちの方向なら動けるだろう？ 89
007-3 敵はおりこうさん？ 92
007-4 最後の仕上げはこうやって 94

巻末資料

●作戦資料101　緊急情報
●作戦資料102　ROMとRAM
●作戦資料103　コントロール・キーの使い方
●キャラクタテーブルⒶ　●キャラクタテーブルⒷ

●カバー・本文イラスト　高橋裕之（ファクトリー）

作戦指令 004

少しうしろをふり返ってみよう

```
1Ø A=1:PRINT A
2Ø RUN
```

と入力してみよう。「RUN」のあとで，リターン・キーを打ち込んでも何もおきないね。つまり，ダイレクトに「RUN」と入力しない限り，コンピューターは「プログラム」を実行しないんだ。もちろん，10行の「PRINT　A」という命令も実行しないことになるね。

それじゃ，このプログラムを実行するとどうなるかな？

```
RUN
```

と入力してみよう。画面には，たてに「1」という数字が次々に出てくるね。ストップ・キーを押さない限り，止まらない。

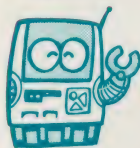
「RUN」とダイレクトに入力したのは，1回だけなのに，コンピューターは何回もプログラムをくり返し実行しているみたいだ？

そう，その通りなんだ。「RUN」と命令されたコンピューターは，10行の命令を実行し，画面に「1」と表示して，次の20行へいく。20行では「RUN」と書かれているプログラムを実行する，ということは，順番にその行に書かれている命令を実行するわけだから，「RUN」という命令を実行して，また，10行へ行くんだ。

ストップ・キーでむりやり実行を止めるまでは，この動作を永遠にくり返し続けているわけだ。

少し頭がこんがらがってきたかな？　ダイレクト・モードでは，一つの命令を実行するには，リターン・キーを押す必要があったね。リターン・キーが命令をコンピューターまで届ける役割をしてたんだ。そして，この「RUN」命令は，プログラムの一つ一つの命令を自動的にコンピューターに送り込む，ベルトコンベアーみたいなもの，と思えばいいね。

ところで，この「RUN」という命令には，もう一つ働きがあるんだ。その前に，このプログラムを一度消してみよう。

NEW

```
NEW
```

とダイレクトに入力してみる。そのあと「LIST」をとっても何にも表示されないね。そう，「NEW」はプログラムを全部消す命令だったね。

さて，

```
A=1……①
```

OK
NEW
OK
LIST
OK

OK
A=1
OK
PRINT A
1
OK
10 PRINT A、
PRINT A
0
OK
A=1:RUN
0
OK

とダイレクトで打ち込んでほしい。次に，

PRINT A……②

と入力してみると，ちゃんと「 1」と表示されるね。

1Ø PRINT A……③

として，

PRINT A……④

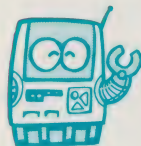
と入力してみよう。今度は「 Ø」になっちゃったね。

A＝1：RUN……⑤

としても「 Ø」のままだね。

最初に変数Aに1を代入して，次に「PRINT」した時には，「A」の値は「 1」だったのに，プログラムを打ち込んだあと（④の時）には「A」の値は「 Ø」になっていた。⑤では，「Aは1だよ。」と命令したあとなのに，「A」は「 Ø」になってしまった。

コンピューターで「A＝1」とするためには，「A」という変数の値がいくつか，ということと，その値をどこに記憶しているか，という事の2つを「RAM」と呼ばれる部分に記録しているんだ。この変数の登録を，いつまでも残しておくと，次々にいろんな変数を使っていると記憶するための場所がなくなってしまう。そこで，「RUN」命令や，プログラムを新しく追加したり，けずったりした時，それまで保存していた変数の値を，全部一度に消す必要があるんだ。

クリアー機能

この働きのことを「CLEAR（クリアー）」機能という。「ホーム・クリアー・キー」が，画面の絵や文字を消すのに対して，この「CLEAR」機能は変数の記憶を消してしまう働きをするんだ。

この「クリアー」機能は，前に出てきた「RUN」やプログラム変更の他にも，次のような命令を使うと自動的に働いてしまうんだよ。

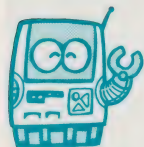
NEW　（プログラム自体も消してしまう。）
CLEAR　（プログラムの中で使い，変数の値をクリアーする。）
LOAD　（他のプログラムを読み込む。）

「RUN」命令は，プログラムの中で使うこともあるけど，ほとんどはダイレクトで使う命令だ。それじゃ，「NEW」はどうだろう。

1Ø NEW

と入力したあと，「RUN」させて，リストをとってみよう。今，せっかく打ち込んだ10行は消えてしまうんだ。いたずら用プログラムでもない限り，「NEW」命令は，あまりプログラム中で使わない方がいいみたいだね。

このように，ダイレクトモードで使う方が多い命令には，どんなものがあるんだろう？

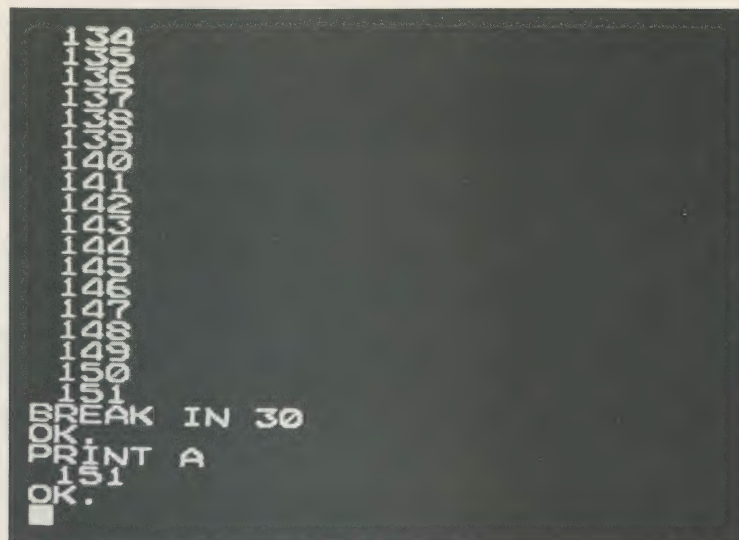
LIST

「LIST」も，その代表的な命令だよ。プログラム中で使えないこともないが，ほとんどは使われないね。

```
LIST              (プログラム全部)
LIST 2Ø—          (20行から最後まで)
LIST —1ØØ         (最初から100行まで)
LIST 2Ø—1ØØ       (20行から100行の間だけ)
```

の4通りの使い方があることは知ってるだろう。

次のプログラムを入力してみよう。

```
1Ø  A=1
2Ø  A=A+1
3Ø  PRINT  A
4Ø  GOTO  2Ø
```

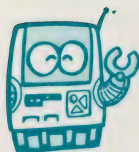
このプログラムを「RUN」すると，画面には数字が一ずつふえなが

ら，次々と表示される。どこかでストップ・キーを押すと，

BREAK IN〜

と表示される。「 〜 」はブレーク・キーを押した時に，コンピューターが実行していた行番号だ。ここで，

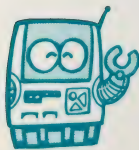
PRINT A

としてみよう。「BREAK 〜 」と書かれた上の行に表示されている数字と同じ値になっているだろう（タイミングによっては一つ多い値）。

さて，今，中止されているプログラムの実行をその場所から再開したい時には，どうすればいいのだろう？ 「RUN」するとまた「 1」から数え始めてしまう。

CONT

CONT

とすると，さっきの数字から表示されるだろう。つまり，ブレーク・キーや，プログラム中の「STOP」という命令で，一度中断されたプログラムを再開するための命令なんだ。

この「CONT」，プログラムを中断したあと，ダイレクト・モードで何かのエラーを出したり，あるいは，プログラムを変更したり，いろんな条件によっては，

?CC ERROR

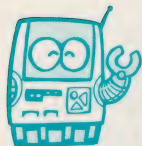
?CC ERROR

「キャント・コンティニュー」つまり，「再開できません。」というエラーが出てしまうんだ。

テープへの書き込みや，読み出しの命令，「LOAD」や「SAVE」，また，書き込みがうまくいったかを確かめる「LOAD　?」という命令も，普通はプログラム中では，ほとんど使われないんだ。また，ゲーム・ベーシック画面からBGグラフィック画面を呼び出すための「SYSTEM」命令も，プログラムには必要ないだろうね。

Ver. 3では，ダイレクト・モードで使う命令がいっぱい出てきて，ふえてくるよ。ジャンプ編で説明することにしよう。

004－2 たった8個の命令で

次のページのリストを見てみよう。「ファミリー・ベーシック　ホップ編」で作ったゲーム・プログラムだ。このプログラム，一体いくつの命令を使ってるんだろう？何となくいっぱい使っているようだけど，実は，全部で8個の命令と4個の関数，それに6個の記号（「AND」とか「＝」とか，演算子なんていうむずかしい名前だったね。）しか使ってないんだ。

①画面に表示するキャラクターを決める。

②それぞれの最初の位置や，得点の最初の値を決める。

③コントローラーからの入力を見て位置を決める。

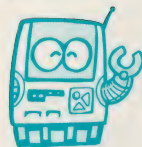
④それぞれのキャラクターを動かす。

⑤当たったかどうかの判定。

基本的には，上の5つのことで作られているんだ。

初期設定

まず，①と②。むずかしい言葉を使うと，**初期設定**とか，初期値を決める，とかいう。ゲームの一番初めにやることで，ゲームが始まっちゃうと，もうここはいらないんだ。ここで使われているのが，「CHR$」と「DEF　SPRITE」。この2つで表示するキャラクターが決まっているんだ。

関数

次に，「STRIG」と「STICK」という2つの**関数**で，コントローラーのどのボタンを押しているかを調べるのが，③の役割だよ。

コンピューターで言う「関数」とは，たとえば，

```
PRINT CHR$(122)
A=STRIG(Ø)
```

の，「CHR$」や「STRIG」のように，ある単語をコンピューターに送ると，その中身を数字や文字で返してくる，その単語のことを言うんだ。もし，君のコンピューターに，「サイフ」という命令を送っ

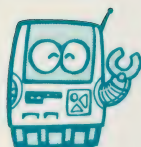
●リスト1

```
10 CLS
20 SPRITE ON
30 DEF SPRITE 0,(0,1,0,0,0)=CHR$(172)+CHR$(173)+CHR$
(174)+CHR$(175)
40 DEF SPRITE 1,(0,0,0,0,0)=CHR$(212)
50 'DEF SPRITE 2,(0,1,0,0,1)=CHR$(162)+CHR$(163)+CHR
$(160)+CHR$(161)
60 DEF SPRITE 3,(0,0,0,0,0)=CHR$(212)
70 DEF SPRITE 4,(0,1,0,0,0)=CHR$(180)+CHR$(181)+CHR$
(182)+CHR$(183)
90 SC=0:SH=3
100 AX=120:BX=120:BY=5
200 IF STICK(0)=1 THEN AX=AX+1
210 IF STICK(0)=2 THEN AX=AX-1
220 IF AX<0 THEN AX=0
230 IF AX>240 THEN AX=240
240 BY=BY+RND(2):BX=BX+RND(9)-4
250 IF BX>240 THEN BX=240
260 IF BX<0 THEN BX=0
270 IF BY>220 THEN BY=5
300 IF FA=1 THEN CY=CY-4
310 IF FB=1 THEN DY=DY+4
320 IF FA=1 AND CY<5 THEN FA=0:SPRITE 1
330 IF FB=1 AND DY>220 THEN FB=0:SPRITE 3
350 IF FA=0 AND STRIG(0)=8 THEN FA=1:CX=AX+4:CY=212
360 IF FB=0 AND RND(6)=1 THEN FB=1:DX=BX+4:DY=BY+16
400 IF FA=1 AND CX+3>BX AND CX+3<BX+15 AND CY+2>BY A
ND CY+2<BY+15 THEN 1000
410 IF FB=1 AND DX+3>AX AND DX+3<AX+15 AND DY+5>220
AND DY+5<235 THEN 1500
500 SPRITE 0,AX,220
510 IF FA=1 THEN SPRITE 1,CX,CY
520 SPRITE 2,BX,BY
530 IF FB=1 THEN SPRITE 3,DX,DY
700 GOTO 200
1000 FA=0:SPRITE 1:SPRITE 2
1010 SPRITE 4,BX,BY
1020 SC=SC+10
1100 GOTO 2000
1500 FB=0:SPRITE 3:SPRITE 0
1510 SPRITE 4,AX,220
1520 SH=SH-1
2000 LOCATE 10,5:PRINT "SCORE =";SC
2010 IF SH=0 THEN 3000
2020 LOCATE 10,8:PRINT "ﾉｺﾘ =";SH
2100 BEEP
2110 FOR I=0 TO 1500:NEXT
2120 BEEP:SPRITE 4
2200 CLS:GOTO 100
3000 LOCATE 10,8:PRINT "GAME OVER !"
3100 END
```

たら，サイフの中身がいくらあるかを教えてくれる機能があるとしよう。

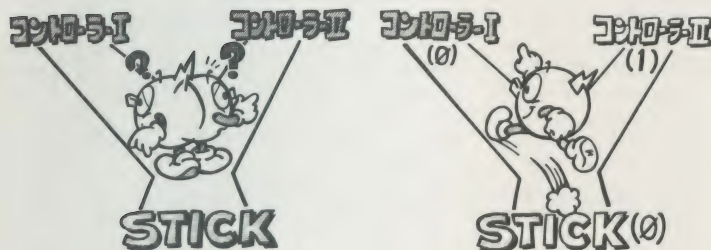
A＝サイフ：PRINT　A

とすれば，「300」とか数字で返ってくるはずだね。(「0」だって？　そんなこと，私は知らないよ！)。この「サイフ」というのも立派な関数といえるんだ。

単語のあとに「( )」がついてるのは，たとえば，コントローラーが2台あって，どっちのコントローラーの情報を知りたいのかによって，コンピューターの処理が違うので，コントローラーⅠなら「(0)」，Ⅱなら「(1)」のように数字を入れて区別するんだ。だから，区別する必要のないものは，単語のうしろの「( )」はいらないことになるね。

FRE

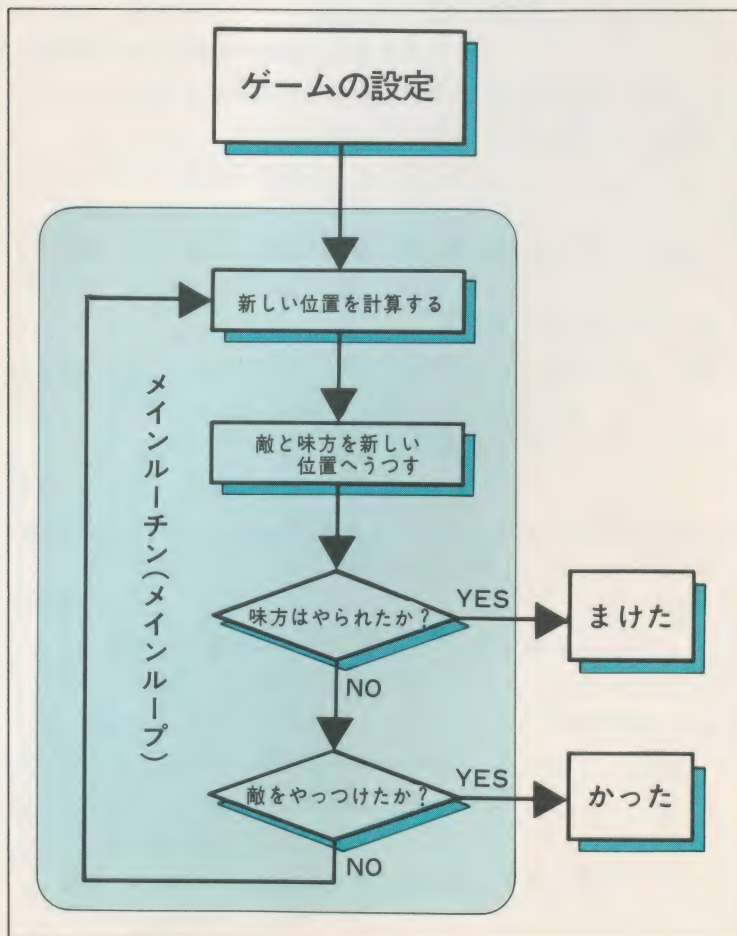
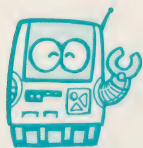
A＝FRE：PRINT　A

この「FRE」という関数は，コンピューターのプログラムや変数を記憶する場所（メモリー）が，あと，どの位残っているかを教えてくれるものなんだ。この場合，残りのメモリーというのは1つしかないから，区別する必要がないので，「( )」はいらないのは，もうわかるだろう？

③ではもう一つ，「STRIG」や「STICK」，「RND」で決めた位置が，画面の外に出てやしないかをチェックしているんだよ。自分のキャラクターが，一番左にいて，コントローラーでもっと左に行

け，という命令を出していたら無視する，というようにね。書いてはいけない場所にむりやり書こうとすると，エラーになってしまう。

そうやって決まった各キャラクターの新しい位置を④で書いてある「SPRITE」という命令で，移動したり，見えなくなるものを消したりしているんだ。

⑤では，レーザーが当たったかどうかを判定して，当たっていなければ「GOTO」で③の処理へ進んでいくわけなんだ。

いかにもむずかしそうな長いプログラムも，こうやって分けて行くと，いくつかのかたまりに分けられる。そうして，一つ一つのか

たまりを見ていくと，そのプログラムが何をしようとしているのかが，わかってくるんだ。

ルーチン

このプログラムのかたまりのことを「**ルーチン**」とも言う。また，③→④→⑤→③→④→⑤のように，くり返して処理することを「**ループ**」と言う。プログラムのワッ！！っていうわけだね。

ループ

①②　**初期設定ルーチン**
③　**入力判定ルーチン**
④　**移動ルーチン**
⑤　**判定ルーチン**

③，④，⑤をメインルーチンまたは，メインループというんだ。この「メイン」とは「主要」という意味だよ。「ルーチン」とか「ループ」という単語は，ぜひ覚えておこう。

004-3 コンピューターに入力するのは

人間が何をしたいのかを，コントローラーから読みとる関数は，「STRIG」，「STICK」という二つがあるのは勉強したね。それじゃ，他にはどんなものがあるんだろう？

Ver. 2のカセットで電源を入れた直後に，「アナタノナマエハ？」と聞いてきたよね。ということは，コンピューターに数字や文字を教えることができるはずだね。

```
10  A$=INKEY$
20  CLS:LOCATE 0,0:PRINT A$
30  GOTO 10
```

無限ループ

このプログラムを入力してみよう。全体の流れはわかるね。A$という文字変数に「INKEY$」というのを入れる。20行では，画面を消して，左上にさっきのA$を表示する。30行では，10行へいく。ストップ・キーを押さない限り，10行から30行までがいつまでも続く，ループ（無限ループ）になってるね。

「RUN」してみよう。画面はまっくらになったね。20行で「CLS」をしてるから当然なんだ。ここで何かキーを押してみよう。文字か数字のキーだよ。

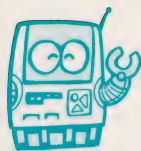
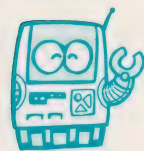
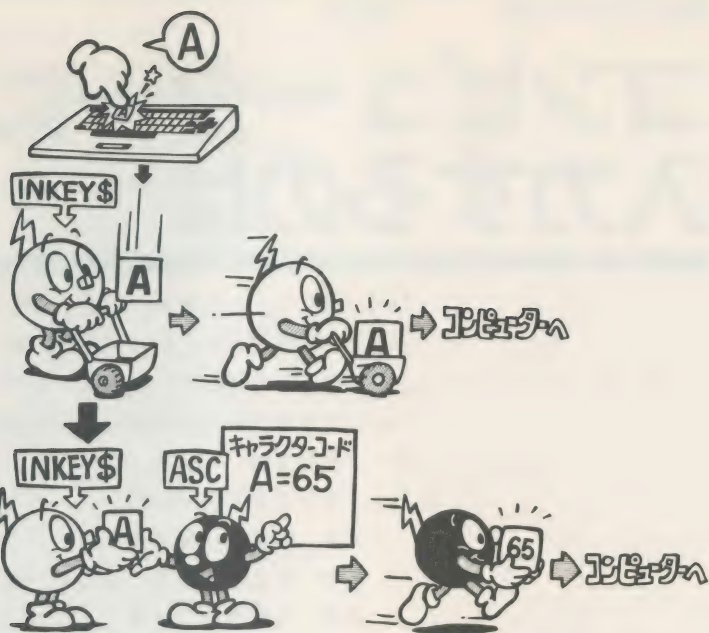
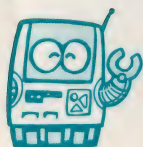
INKEY$

すると，何か字が現れたみたいだね。でもすぐ消えちゃった。しかも，現われた文字は，押したキーと同じ字だったみたいだな。

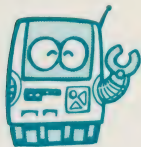
この「INKEY$」というのは，コンピューターが「INKEY$」と書かれた行を実行している時に何かキーが押されていたら，そのキーと同じ文字を返してくる文字関数なんだ。

ここで，20行を変えてみよう。

```
20  CLS:LOCATE 0,0:PRINT ASC
    (A$)
```

ASC

このままRUNして，何かのキーを押すと，いろんな数字が出てくる。「A」というキーを押すと「65」と出るはずだ。**キャラクターコード表B**を見てみよう。「A」のコードは「65」だ。つまり，「ASC」という命令で押したキーのコードナンバーを表示するプログラムなんだ。

ここで，キーボードのカーソル・キーを押してみよう。「▲」のキーを押すと，「30」と表示されるだろう。ところが，キャラクタコード表Bには「▲」なんてのってなくて，Bコードナンバー30は，キャラクタテーブルBのD列の6番目だって書いてある。

これは，普通のパソコンと違って，ファミコンではいろんなキャラクタをコンピューター内部に持っているため，コードナンバー0から31の間は2通りの使い方をしているせいなんだ。誰かがパソコンをもっていたら，そのマニュアルのキャラクタコード表を見てもらおう。コードナンバー30（10進法の）は「↑」となっているパソコンがほとんどだ。

特殊キーのコード

この「INKEY$」は，あとで説明する命令と違って「▲」などのカーソル・キーや「INS」，「DEL」キーなどの**特殊キーが**押されたかどうかもわかる命令なんだ。この命令を使うと，マリオなどのキャラクターをコントローラー抜きで，キーボードのカーソル・キーで動かすことができるんだよ。

```
1Ø  AX=Ø:BX=Ø
2Ø  IF STICK(Ø)=1 THEN
    AX=AX+1
3Ø  IF INKEY$=CHR$(28)
    THEN BX=BX+1
4Ø  PRINT AX,BX
5Ø  GOTO 2Ø
```

このプログラムは，コントローラーの「✚」を右へ押すとAXが，キーボードの「▶」を押すとBXが，1づつふえていく，というものなんだ。キーボードがコントローラーの代わりをできるのがよくわかるだろう。

```
「◀」⇨CHR$(29)
「▲」⇨CHR$(3Ø)
「▼」⇨CHR$(31)
```

いろんな特殊キーをためしてみよう。

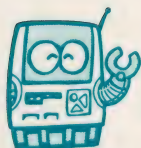
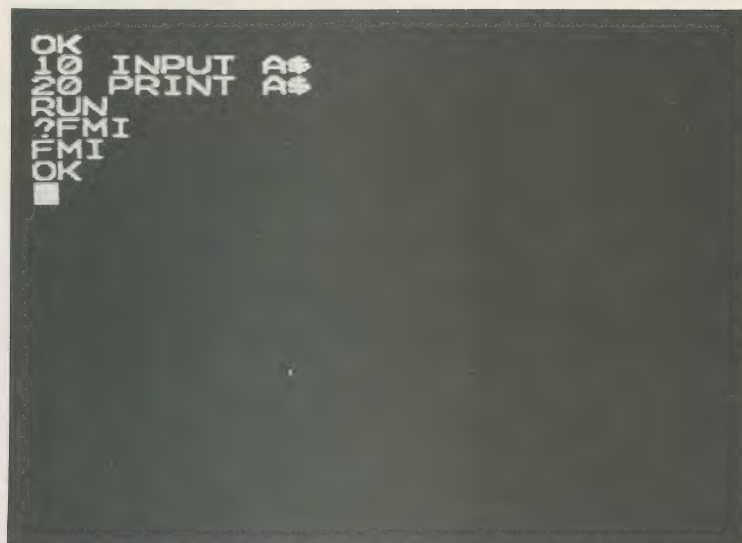
㊟ ページのプログラムを実行している時にファンクションキーを押すと，そのファンクションキーに登録されている文字が1字づつ表示される。これは，ファンクションキーを押すことで，一度に文字が並んで行列まちをしていると思ってほしい。

パソコンによっては，この「INKEY$」が現われるまでに押したキーを記憶していて，この行列まちのような形で覚えているもの（キー・バッファという）もある。

INPUT

では，2字以上の文字や数字を入力するには，どうすればいいんだろう？前のプログラムを「NEW」して消してから，次のプログラムを入力しよう。

```
1Ø  INPUT A$
2Ø  PRINT A$
```

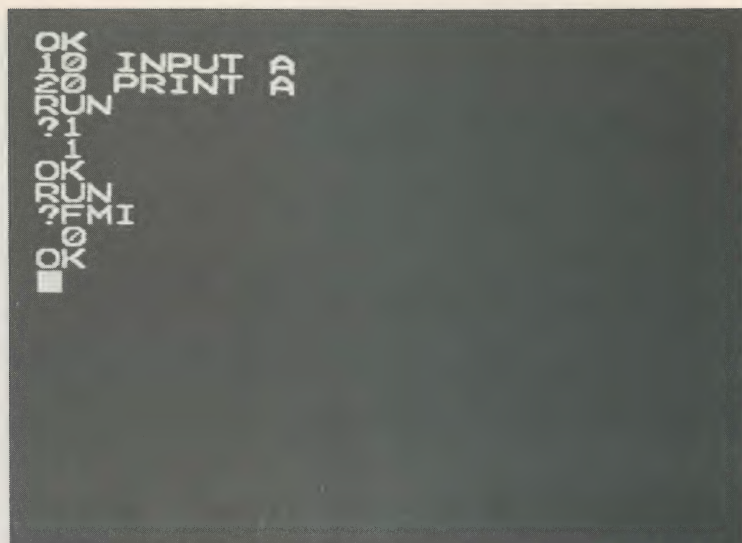
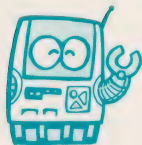
「RUN」すると「？」マークが出て，すぐうしろにカーソルが点

滅しているね。

「F」,「M」,「I」と一つづつキーを押してみよう。一字づつ画面に表示され,「I」のうしろにカーソルが移っただろう。ここでリターン・キーを押すと,次の行に,もう一度「FMI」と表示されたね。

2回目の「FMI」はわかっただろう。20行の「PRINT　A$」という命令だね。ということは,A$の中身は「FMI」になっている。つまり,10行の「INPUT　A$」は,「A$」の中身は何にしましょう?とコンピューターから君に問いかけるようにしろ,という命令だったんだ。

「INPUT」という命令を受けとると,コンピューターはまず「?」というマークを出し,うしろにカーソルを点滅させることで,早く入力して,と要求してくるんだ。

もう一度「RUN」して,今度は「▼」と「K」を一回づつ押してみよう。「▼」キーを押すとカーソルは一段下がって,下に「K」を表示した。でも,リターンを押すと,A$には「K」の一文字しか入っていないことがわかる。つまり,「INKEY$」と違って特殊キーは受けつけないのがわかるよね。

「RUN」したあと,何でもいいから文字キーを32個以上押したあと,リターンキーを押すと,

?ST ERROR

?ST ERROR IN 1Ø

となる。ホップ編で話した文字列の長さ制限を覚えてるかな？このエラーは「ストリング・トゥー・ロング」エラーといって，文字列の長さが長すぎるよ，というエラーなんだ。

```
1Ø INPUT A
2Ø PRINT A
```

と変えてみよう。前の命令では「A$」，今度は「A」。これは数値を入　力しろという命令なんだ。ためしに「1」のキーを押すと「　1」と表示される。前のプログラムの時では「1」を入　力すると「1」と表示されたはずだね。違いがわかるかな？前のプログラムでは「1」は文字としてあつかったため，そのまま「1」と表示されたけど，今度は数字としてあつかうために，「1」の手前に「+」を書いてあるためなんだ。〔(「+」の場合は省　略して代わりに，「　」(スペース) を置く約束だったよね。)〕

「F」,「M」,「I」と押してリターンキーを押してみよう。数値変数の中身を聞く命令に文字列を入　力するんだから，ほとんどのパソコンでは「Type　Mismatch」(タイプが違うよ！) エラーになるけど，ファミコンではエラーにならないで，何も入　力しない(つまり Ø) あつかいになるようだね。

```
OK
10 INPUT "Xハ";A
20 PRINT A
RUN
Xハ?■
```

　　10　INPUT　〝Xハ〞; A

と変えてみよう。

　　Xハ?

となって「?」のうしろにカーソルが点滅している。

　プログラムの中でいくつも「INPUT」文があると，入力を要求されたときに，何を入力すればいいのか，わからなくなるときがある。そんなとき，「INPUT」のあとに「〝」(ダブルクォーテーションマーク) で囲んだ文字列を書いて，「;」(セミコロン) を置くと，まず文字列の中を表示したあと，入力を要求してくるんだ。

;

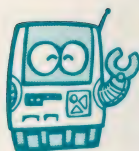
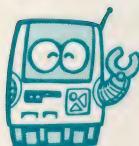
　　10　PRINT　〝Xは〞; : INPUT　A

というのと同じ働きなんだ。

　今度は，

　　10　INPUT　〝X=〞, A

としてみよう。今度は「X=」のうしろで，直接カーソルが点滅しているね。「?」のマークが出ないようにするには，文字列のあとの「;」を「,」(カンマ) に変えるだけでいいのだ。ただ，この使い方は，必ず文字列がないと使えないよ。つまり，この10行を書きかえて，

?と，

　　10　PRINT　〝X=〞; : INPUT　, A

ではだめて，

```
1Ø PRINT 〝X=〃;:INPUT 〝〃
  ,A
```

としなければいけない。

なぜだろう？この「，」は，実はいくつもの変数を入力するときの区切り記号にも使うんだ。

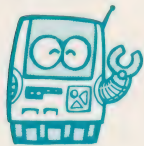
区切り記号

```
1Ø INPUT 〝A,Bは〃;A,B
2Ø PRINT A*B
```

としてみよう。「RUN」したあと，「3」キーとリターンキーを押してみよう。あれ，もう一度「?」が出てきたね。「4」キーとリターンキーを押すと，「12」と表示される。

```
PRINT A:PRINT B
```

としてみると，「A」には「　3」，「B」には「　4」が入っている。10行の命令は一度に「A」，「B」2つの変数の値を入力せよ，という意味だったんだ。だから，最初に「3」を入力しても，もう一個いるから，もう一度「?」のマークを出して「B」の方の値の入力を要求してきたわけなんだ。

もう一度「RUN」して，今度は「2」，「，」，「5」，リターンキーと押してみよう。「　1Ø」になっただろう。入力する時も「，」で

```
OK
10 INPUT "A,Bハ";A,B
20 PRINT A*B
RUN
A,Bハ?3
?4
 12
OK
```

区切ることによって，2個の数字を一度にできるんだ。

この「，」で区切ることで2個以上，それも，文字も数字も入力できるんだ。

INPUT A，B$，C

という命令なら「3」，「，」，「T」，「E」，「，」，「4」と入力すると，ちゃんとそれぞれの変数に，押したものが入っている。

```
1Ø INPUT "A，Bハ  "；A，B
2Ø PRINT "A＋B＝"；A＋B
3Ø PRINT "A－B＝"；A－B
4Ø PRINT "A＊B＝"；A＊B
5Ø PRINT "A／B＝"；A／B
```

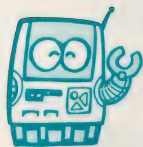
このプログラム，もう説明しなくても，君にはわかるよね？

最後に，じゃあ「ファミ，コン」というように「，」をふくんだ文字列は絶対入力できないじゃないか！と気がついた君。エライ！！その通りだ。そのかわり，

```
1Ø LINPUT A$
2Ø PRINT A$
```

というプログラムを実行してみよう。ちゃんと「，」の入った文字列も一つの文字列として入力できたね。

```
1Ø LINPUT "TEST＝"；A$
```

としてみよう。「A」と入力すると，今度は，A$は「TEST＝A」という文字列になっている。この「LINPUT」のあとに文字列をつけると，その文字列と，入力された文字列をたしたものが文字変数に代

LINPUT

```
OK
10 LINPUT "TEST=";A$
20 PRINT A$
RUN
TEST=■
```

入される，という仕組みになっているんだ。

だから，この「LINPUT」は，数値変数の入力や２つ以上の変数の入力には使えないことになるね。つまり，文字変数だけの入力用なんだ。また，この命令を使うと「？」は表示されない。だから「〝〟」で囲んだ文字列のうしろは，「；」でも「，」でも働きはまったく一諸なんだ。

作戦指令005

ベーシック自由自在

005−1
あっちの道、こっちの道

ファミコンの「メモリー」が少ない，ということは，ゲーム作りには一番の難関になるのはわかるだろう。わずか1982バイトの中で，よりおもしろいゲームを作るためには，いろんなテクニックを使わなければならないんだ。

```
10 DEF SPRITE 0,(0,1,0,0,0)=CHR$(172)+CHR$(173)+CHR$(174)+CHR$(175)
90 DEF SPRITE 4,(0,1,0,0,0)=CHR$(180)+CHR$(181)+CHR$(182)+CHR$(183)
```

このプログラムは，ホップ編のゲームリストの一部だね。よく見ると，10行と90行で違うのは「DEF　SPRITE」の次の数字が「0」と「4」というのと，「CHR$」のカッコの中だけだ。

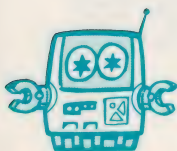
このプログラムを打ち込んだところで，ダイレクトに

```
PRINT FRE
```

と入力してみよう。「FRE」はメモリーの残りを表わす関数だったよね。「1860」になったはずだね。とすると，この2行のプログラムで「122」バイト（メモリーの単位をバイトという。）使ったことになるね。

GOSUB

さて，「NEW」したあと，次のプログラムを打ち込んでみよう。

```
10 A=0:B=172:GOSUB 5000:A=4:B=180:GOSUB 5000
5000 DEF SPRITE A,(0,1,0,0,0)=CHR$(B)+CHR$(B+1)+CHR$(B+2)+CHR$(B+3):RET
```

URN

この状態で「FRE」をしらべると，「1878」になるだろう。つまり，こっちのプログラムでは「1Ø4」バイトしか使ってないわけだ。

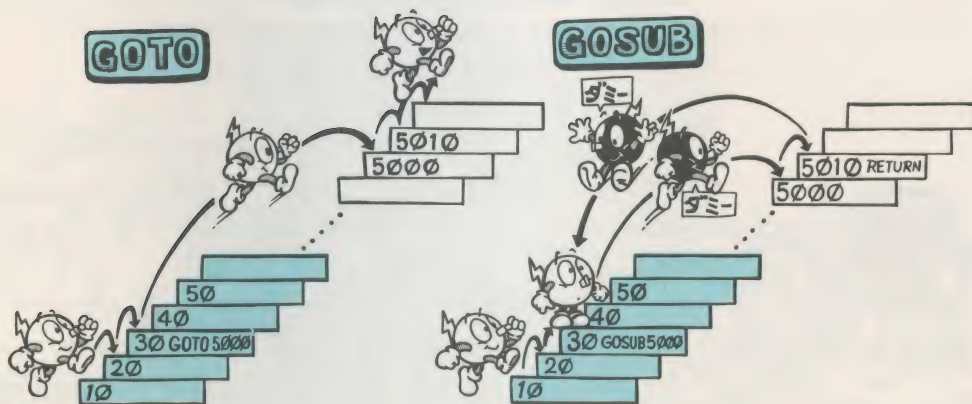
使うメモリーが少なくなったのはいいけど，さて，この2つのプログラムは同じ働きをするのかな？

```
1ØØ SPRITE Ø,1ØØ,1ØØ
11Ø SPRITE 4,15Ø,1ØØ
12Ø END
```

上のプログラムをそれぞれのプログラムにつけ加えて「RUN」すれば，同じ働きをすることがすぐわかるね。

じゃあ，2番目のプログラムはどういう働きをしているんだろう？

10行で変数A，Bにそれぞれ0，172を代入したあと「GOSUB 5ØØØ」と書いてある。今まで「GOTO」という命令が出てきたことはあったけど，これは初めてだね。

コンピューターがプログラムの上を歩いている人としよう。その人が歩いている足の下にある命令を実行しているというわけだ。こ

の人が「GOTO」の命令に会うと，そのあとに書かれた行番号の所へ飛んでいってしまう。「GOTO　1000」と書かれていたら，今いる所から，行番号1000に書かれている命令の上までジャンプしていくわけなんだ。

それに対して「GOSUB」はどうだろう？　行番号100に「GOSUB　1000」という命令があると，1000行の命令から実行しはじめる。しかし，途中で「RETURN」という命令に行きあうと，もとの「GOSUB　1000」という命令の次に書かれた命令の所に帰ってくるんだ。

RETURN

言いかえれば，コンピューターは行番号100で「GOSUB」命令に会うと，本体はそこに残したまま，ダミー（代理）を1000行に飛ばす。そして「RETURN」に会うと，そのダミーは本体の所へ帰ってきて，また，本体が歩き始めるわけなんだ。

このように「GOSUB」命令は，出て行った所から必ず「RETURN」命令で帰って来なければいけないんだ。

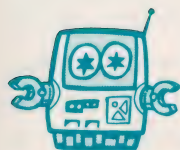
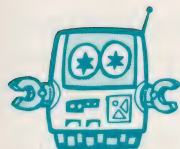
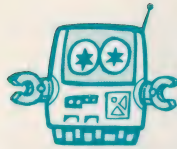
```
10  I=0
20  GOSUB  100
30  END
100  I=I+1:GOTO  20
```

このプログラムを「RUN」すると，すぐ「？OM　ERROR」になる。メモリー不足エラー，という意味だね。ところが，「PRINT　FRE」とすると，「1938」バイトも残っているよ，と表示されるね。

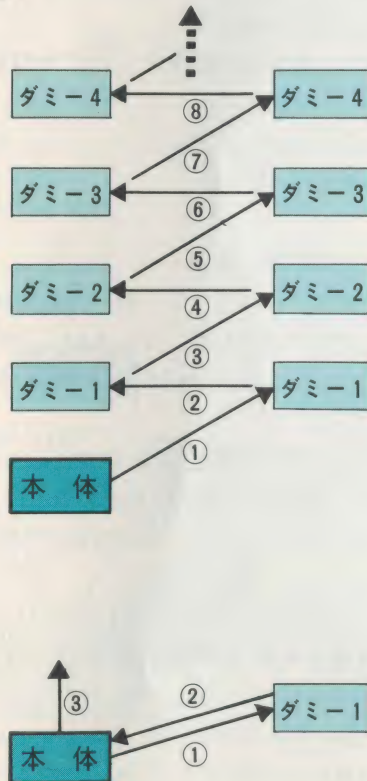
プログラムを見ると，20行から「GOSUB」命令で100行へいったのに100行からは「RETURN」命令でなく「GOTO」命令で20行へいっている。そして，その20行から，また「GOSUB」で100行へいくことをくり返しているね。

どこから出発したか

コンピューターが「GOSUB」命令でどこかへ行くときには，返ってくるためにも，**どこから出発したか**（GOSUB命令がどこに書かれていたか）を記憶する必要があるんだ。ところがこのプログラムでは何回も「GOSUB」をくり返しているため，出発点を覚える場所がなくなっちゃったんだ。ためしに「PRINT　I」をしてみよう。29回くり返したことがわかるだろう。

```
CLEAR
PRINT FRE
```

としてみると，1942バイトだね。(変数Iをクリアーしたから4バイトふえた。)とすると，この出発点を覚えているメモリーは，僕達の使える部分とは別に，コンピューターが持ってるのがわかるね。さっきの「OM」エラーとは，その部分がメモリー不足になったよ，という意味だったんだ。

サブルーチン

このように，「GOSUB」で行って「RETURN」で帰ってくる間（ダミーが実行している部分）を「**サブ・ルーチン**」と呼ぶんだ。前のプログラムの5000行がそれにあたるわけだね。このサブルーチンの中から別のサブルーチンへ行ったりすることもできるんだよ。図1を見てみよう。「サブルーチンへ行くことを，ふつう，「GOTO 1ØØ」が「100行へいく」に対して，「GOSUB　1ØØ」は「100行の サブルーチンを呼ぶ」または，「サブルーチンコール」と言う。

●図1

これで2番目のプログラムの役割がわかったよね！　120行の「END」はどうだろう。もしこの「END」がなかったら，コンピューターの動きは次のようになってしまう。

10行→5000行→10行→5000行→100行→110行→5000行

そして，ここで「? RG　ERROR」(＝リターン・ウィズアウト・ゴーサブ)，つまり，「GOSUB」命令を受けてないのに「RETURN」があるよ，というエラーが出てしまう。サブルーチンをプログラムのうしろの方に置くときによくやるエラーなんだ。プログラムのおわりには，必ず「END」をつけるくせをつけようね。

このように，少し気をつけなきゃならない点はあるけども，サブルーチンは使い方さえうまくすれば，プログラムを小さくするのに

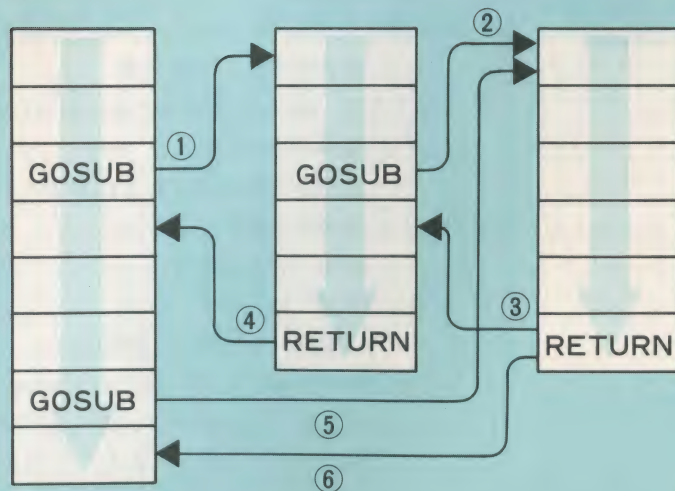
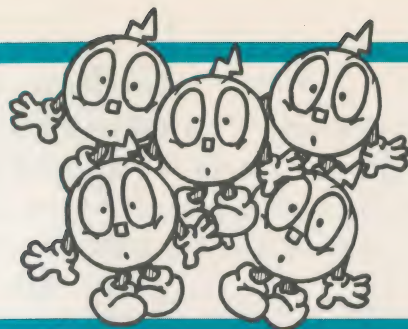
役立つし，わかりやすくプログラムを組むことだってできるんだよ。今回の例では，わずか18バイトしか小さくならなかったけど，同じ処理をくり返すほど，サブルーチン形式にした方が使うメモリーはより少なくてすむんだ。

㊟ 必しも，「GOSUB」1個に「RETURN」1個が必要というわけではない。

例えば，図2 では，「GOSUB」3個に「RETURN」2個だし，例えば，

```
10  A=0:GOSUB 100
20  END
100  IF A=1 THEN RETURN
110  A=A+1:RETURN
```

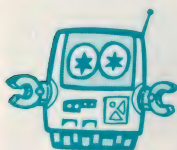
というプログラムでも間違いではないんだ。命令の流れで「GOSUB」で行なった所からは必ず「RETURN」に行き会えればそれでよいのだ。

●図2

005−2 くり返して何回も

§1　思い出したくないテストの点数

ホップ編のプログラムのうち，

```
2110 FOR I=Ø TO 15ØØ:NEXT
```

というのを覚えているかな？　この行はただの時間待ちの行だと説明したよね。

PAUSE

```
211Ø PAUSE 7Ø
```

これでも大体同じくらいの時間待ちをしてくれるんだ。

「PAUSE」という命令には2つの使い方があるんだよ。

```
PAUSE
```

とだけすると，何かのキー入力があるまでは，ずっと待っているし，

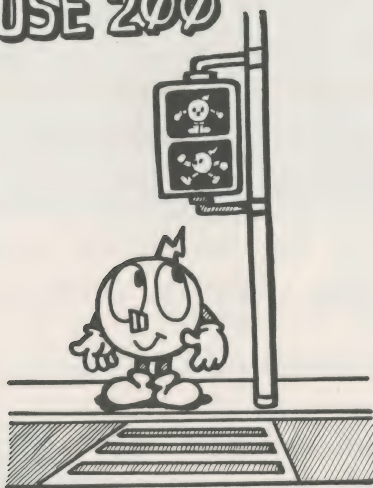
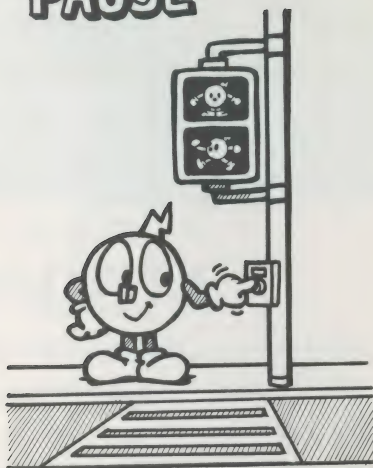
```
PAUSE 1ØØØ
```

とすると，約11秒間待って次の命令に進む。

```
1Ø BEEP
2Ø PAUSE 1ØØØ
3Ø BEEP
4Ø PAUSE
5Ø BEEP
```

「PAUSE」のあとに続く数字はØ～32767の間が許されている。それ以外の数字だと「？IL　ERROR」になってしまうんだ。

じゃあ，もとのプログラムの2110行では約1秒弱の間，何をしてたんだろう？

実は，変数IにØから1つずつ15ØØまで15Ø1回代入していたんだ。おふろに入ってる時，1ØØまで数えて，っていうあれと同じことをしてたんだなー。

それじゃあ，この命令は何の役にも立たないじゃないか。ところがどっこい，すごく役に立つ命令なんだ。

ここに，君の国語と社会と理科の過去3回分のテストがあるとしよう。

	国語	社会	理科
1回目	60	70	50
2回目	30	40	80
3回目	90	100	20

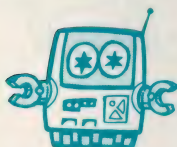
各回の平均点と科目別の平均点を出すには，どんなプログラムを作ればいいんだろう？

●リスト1

```
100 K1=60:K2=30:K3=90
110 S1=70:S2=40:S3=100
120 R1=50:R2=80:R3=20
200 PRINT "コクゴ=";(K1+K2+K3)/3
210 PRINT "シャカイ=";(S1+S2+S3)/3
220 PRINT "リカ=";(R1+R2+R3)/3
300 PRINT "1カイメ=";(K1+S1+R1)/3
310 PRINT "2カイメ=";(K2+S2+R2)/3
320 PRINT "3カイメ=";(K3+S3+R3)/3
```

K，S，Rはそれぞれ国語，社会，理科の頭文字で，そのあとの，1，2，3は1回目，2回目，3回目の略だ。このプログラムはたしかに間違っていない。

でも，もしこれが10回分のテストの結果という事になるとどうなるだろう？

K1とK10

100　K1=60：～：K10=80

ファミコンでは変数の名前は最初の2文字で判断していたから，「K1」と「K10」は同じ変数と見られてしまうね。じゃあ，名前をかえて，

100　KA=60：～：KJ=80

全部のプログラムをこの調子で書くのは，ちょっと大変だし，プログラムが大きすぎちゃって，メモリーがたりなくなっちゃう。そこで新しい考え方をしてみよう。

●リスト2

```
100 DIM K(3),R(3),S(3),T(3)
110 FOR I=1 TO 3
120 READ K(I),R(I),S(I)
130 NEXT
200 DATA 60,70,50,30,40,80,90,100,20
300 FOR I=1 TO 3
310 KH=KH+K(I)
320 SH=SH+S(I)
```

●リスト2

```
330 RH=RH+R(I)
340 T(I)=K(I)+S(I)+R(I)
350 NEXT
500 PRINT "ｺｸｺﾞ=";KH/3
510 PRINT "ｼｬｶｲ=";SH/3
520 PRINT "ﾘｶ=";RH/3
600 PRINT "1ｶｲﾒ=";T(1)/3
610 PRINT "2ｶｲﾒ=";T(2)/3
620 PRINT "3ｶｲﾒ=";T(3)/3
```

§2 READとDATA

新しい単語がいくつも出てきたね。「DIM」とか「READ」とか「DATA」，それに「K (3)」とか「R (3)」とかは，何だろう？

```
10 A=0
20 READ A
30 PRINT A
40 DATA 5
```

DATA

このプログラムを「RUN」してみよう。「 5」と表示されたね。

「DATA」というのは，「資料」という意味の英語「DATUM」の複数形なんだ。データベースとかよく聞くと思うけどその「データ」なんだよ。

READ

「READ」は「読む」という意味だね。とすると，20行は，データを読みにいって，そのデータを「A」に入れる役割をしているんだ。

READ (DATA):A=(DATA)

とも書ける役割をしている。

```
10 READ A,B,C$
20 PRINT A:PRINT B:PRINT C$
30 DATA 3,5,"TEST"
```

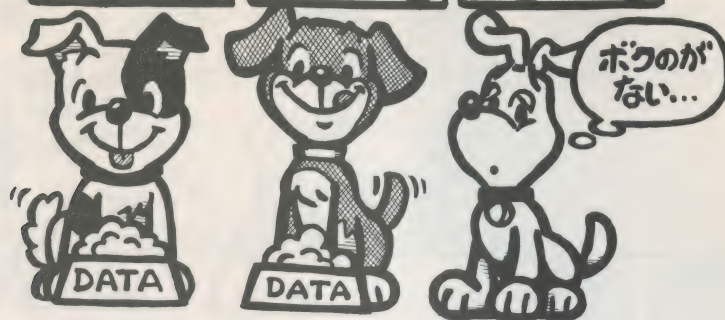
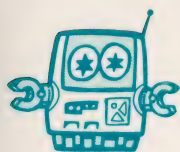
ちゃんと「A」には「3」，「B」には「5」が入ってるのがわかる

だろう。しかも「C$」には，ちゃんと「TEST」の文字が入っている。「READ」命令があるとコンピューターは「DATA」と書かれている行を探しにいって読み込む。次の「READ」のときは，さっき読んだ「DATA」の次のものを読み込みに行く。したがって「DATA」が書かれた行は，「READ」文の前にあってもいいし，また，「READ」文と「DATA」文が何個所にわかれていても，基本的に「READ」と「DATA」の数さえ合っていれば，それでいいわけなんだ。

じゃあ，「DATA」の数が合ってない時はどうなるのかな？

```
1Ø READ A,B
```

●リスト3

```
100 DATA 10
200 READ A,B
300 READ C
400 DATA 3,2,1
500 READ D,E
600 DATA 6
```

```
2Ø DATA 1
```

?OD ERROR

この場合は「? OD　ERROR」(アウト・オブ・データ)つまり，データの数がたらないよ，というエラーが出てしまうんだ。

データの方の数が多い場合には，エラーにはならないけど，ふつうは，必要以上にデータをわざわざ書くことはないだろうね。

```
1Ø READ A$,B
2Ø DATA 1,1
```

この場合，文字変数「A$」は「1」を文字としてとり入れ，数値変数「B」は数値として「　1」をとり入れる。ということは，文字変数用の「DATA」は「〃」マークで囲まなくてもいいことになるね。

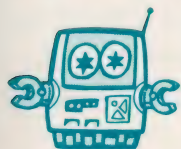
```
2Ø DATA TEST,1
```

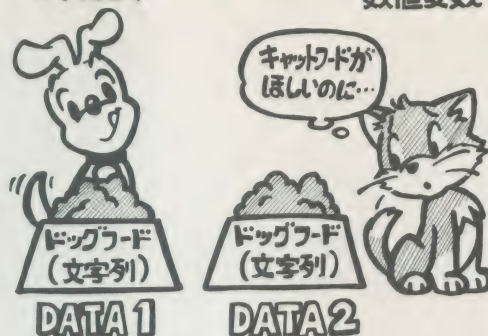
でもいいわけだ。じゃ，

```
2Ø DATA TEST,TEST2
```

としてみよう。

?SN ERROR

```
?SN ERROR IN 1Ø
```

となったはずだね。これは，数値変数「B」を読みにいったのに，そこには数値でなく「TEST2」という文字列しかない，というエラーなんだ。ほんとうは「DATA」文の書きまちがいかも知れないのに，エラーとしては必ず「READ」文のある行のものとしてあつかわれるので，要注意だぞ！

このように，「READ」文の中での文字変数と数値変数との順番と「DATA」文の中での文字列と数値との順番とは，いつも同じでな

ければいけないんだ。

もう一つ，ついでに新しい命令を覚えちゃおう。

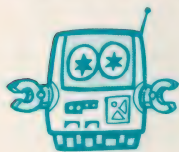
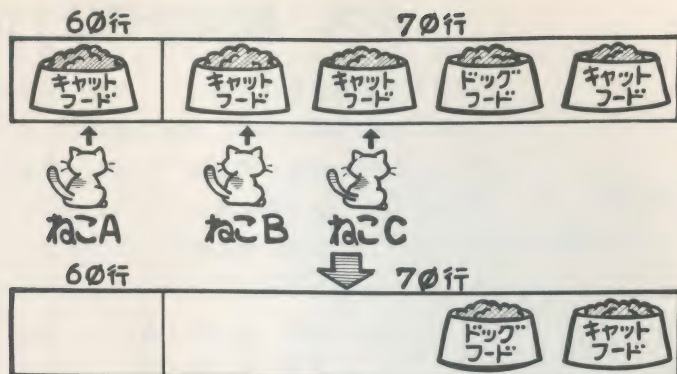
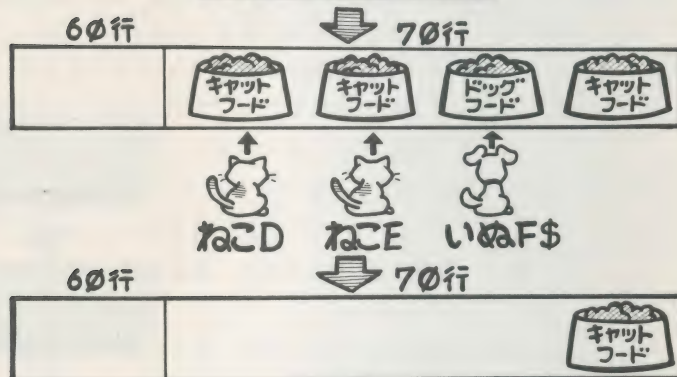
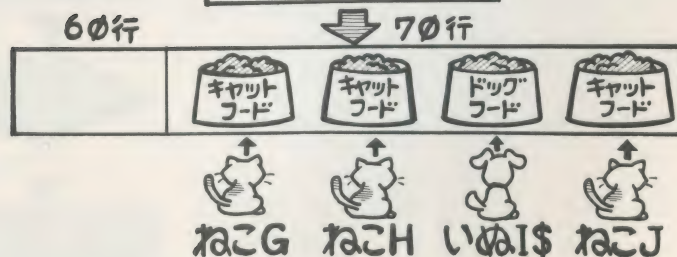
```
1Ø  READ  A,B,C
2Ø  RESTORE  7Ø
3Ø  READ  D,E,F$
4Ø  RESTORE  7Ø
5Ø  READ  G,H,I$,J
6Ø  DATA  4
7Ø  DATA  1,2,TEST,3
```

このプログラムで，「B」，「D」，「G」は1，「C」，「E」，「H」には2，「F$」，「I$」にはTEST，「J」に3が，「A」に4が入る。

RESTORE

「READ」文が1回あるごとに，「DATA」文の何番目を読むか，ということをコンピューターは覚えてるんだけど，この「RESTORE XX」という命令があると，XX行からあとにある，「DATA」文から読み込みはじめる。というわけなんだ。当然，この「XX」が存在しない行だと「？UL　ERROR」（アンディファインド・ライン・ナンバー）そんな行，みつからないよ，というエラーになるし，「XX」行からうしろに「DATA」文がなければ「？OD　ERROR」になってしまうんだ。

㊟　作戦資料001　バグ情報を見て！

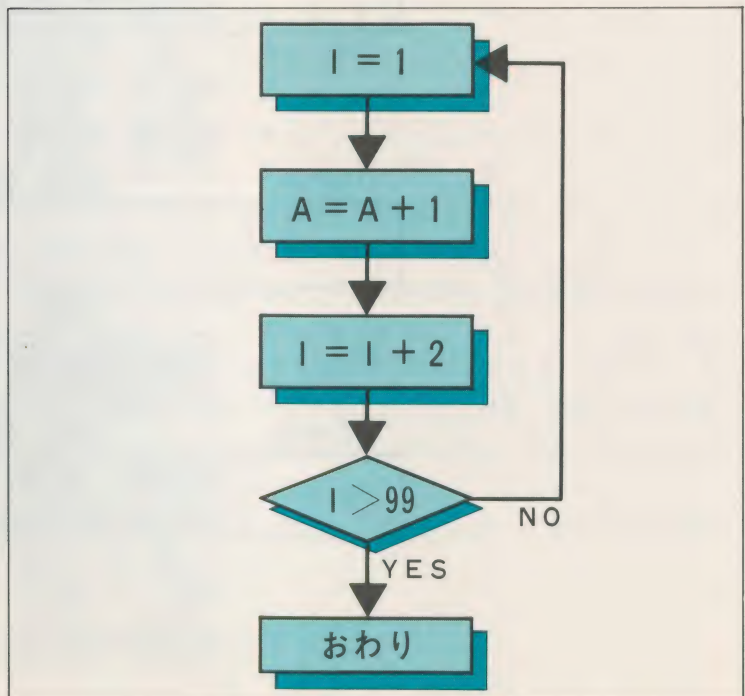
6Ø行
7Ø行
キャットフード
キャットフード
キャットフード
ドッグフード
キャットフード
ねこA
ねこB
ねこC
6Ø行
7Ø行
ドッグフード
キャットフード
RESTORE 7Ø
6Ø行
7Ø行
キャットフード
キャットフード
ドッグフード
キャットフード
ねこD
ねこE
いぬF$
6Ø行
7Ø行
キャットフード
RESTORE 7Ø
6Ø行
7Ø行
キャットフード
キャットフード
ドッグフード
キャットフード
ねこG
ねこH
いぬI$
ねこJ

§3　配列変数とDIM

リスト2にもどろう。リスト1とくらべてみると，どうも「K(3)」というのは，K1，K2，K3のことらしい。

```
1Ø  FOR  I=1  TO  3
2Ø  A=A+1                    ………Ⓐ
3Ø  NEXT
```

この命令は，

```
1Ø  A=A+1
2Ø  A=A+2                    ………Ⓑ
3Ø  A=A+3
```

というのと同じ意味をもつ。「　1」づつふえていく数をたしてるんだ。ところが，1から3までだと同じ3行ですむけど，1から100までだとⒷの形では100行にもなる。けれども，Ⓐの形式では同じ3行ですむ。

```
1Ø  I=1
2Ø  A=A+1                          ………©
3Ø  IF  I<3  THEN  I=I+1
        :GOTO  2Ø
```

としても，Ⓐと同じプログラムにある。つまり，「FOR 〜 TO 〜 NEXT」という文型は，ある数字，この場合は「I」(これを制御変数という。)を，ある値から，ある値まで変化させ（この場合1から3）て，いろんな処理を行なう命令なんだ。

```
1Ø  FOR  I=1  TO  99  STEP  2
2Ø  A=A+1
3Ø  NEXT
```

これは，「 1」から「 99」までの奇数だけの数を，たす，というプログラムだ。「STEP 2」というのは「I」を2つずつふやしていく，という意味だ。つまり，

```
FOR  I=1  TO  3
```

というのは，

```
FOR  I=1  TO  3  STEP  1
```

の省略形だったんだ。「STEP」をつけなければ「STEP 1」のことと一緒のことなんだよ。当然，

```
FOR  I=8Ø  TO  2  STEP  -2
```

のようなのもできるんだ。

```
FOR  I=1Ø  TO  5  STEP  2
```

はどうだろう。まず「I=1Ø」として処理を行なう。「NEXT」と出会うと「I=12」になる。ところが「TO 5」とあって，「I>5」だから，くり返しはしないで次の命令にうつる。つまり，エラーにはならない。(©にあてはめてごらん。)

リスト2がわかってきたかな？「FOR NEXT」で「I」は1から3まで変化する，とすると「K (I)」というのは，K (1)，K (2)，K (3) と3回出てくるわけだ。リスト1のK1，K2，K3と一緒だね。

図3を見てみよう。この「K ( )」のように書くと，メモリーが許すかぎり，いくつふえても同じ「K」という名前の変数の仲間として使えるんだ。

リスト１のやり方では，回数がふえてくると，どうしようもないのは前に言った通りなんだけども，この「( )」を使った変数のやり方だと，たとえば，回数が1Ø回なら，

```
11Ø FOR I=1
 TO 1Ø
```

と直して，あとは200行のDATA文をそれに応じてふやすだけでいいんだ。

●図３

配列変数

このような「( )」を使った変数を「**配列変数**」というんだ。ただ，この配列変数を使うときは，あらかじめ，「こんな配列変数を使いますよ。」とプログラムのはじめでコンピューターに対して，「宣言」しておかなければいけないんだ。

DIM

「DIMENSION」，容積とか大きさ，次元という意味の略語の「DIM」というのが，その，宣言文だったんだ。

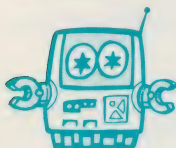
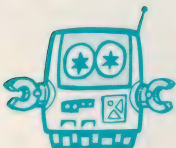
```
DIM K(3)
```

これは，「K」という文字であつかう配列変数を，「K(Ø)」から「K(3)」まで使いますよ，という意味なんだよ。（カッコの中の数字を添字という。）

この「DIM」で宣言してない配列変数を使ったり，宣言している大きさより大きい添字を使ったりすると，

```
?IL ERROR
```

になってしまう。つまり回数が10回になった時は，100行も，

```
1ØØ DIM K(1Ø),R(1Ø),S(1Ø),T
 (1Ø)
```

に直す必要があるわけだ。

リスト2の「T (3)」は各回の合計，KH，SH，RH はそれぞれ国・社・理の合計だってことはわかるよね。リスト2では2回も「FOR　NEXT」ループを使っている。これを短かくすると，

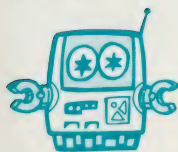
●リスト4

```
100 DIM K(3),R(3),S(3),T(3)
110 FOR I=1 TO 3
120 READ K(I),R(I),S(I)
130 KH=KH+K(I)
140 SH=SH+S(I)
150 RH=RH+R(I)
160 T(I)=K(I)+S(I)+R(I)
170 PRINT I;"カイメ=";T(I)/3
180 NEXT
200 DATA 60,70,50,30,40,80,90,100,20
500 PRINT "コクゴ=";KH/3
510 PRINT "シャカイ=";SH/3
520 PRINT "リカ=";RH/3
```

になる。いずれにしても，リスト1よりはメモリーをたくさん使うけど，あつかうデータの量が多くなると，今度はリスト2や4の方が，断然小さくなってくるのはわかるよね。

ところで，ページでも言ったように，テストの点数はたてと横に書ければわかりやすい。配列変数も図3　だけじゃなく，たてと横に使えないんだろうか？

図4　をみてみよう。

2次元

ファミコンでは，配列変数は2次元まで許されている。1次元というのは，図3　のような，たて，または横だけの配列変数なんだ。このように，たてと横の2つの要素をもったものを2次元というんだ。カーソルの位置を表わすのも，X座標・Y座標という2つの要素が必要だったよね。あれと同じっていうわけなんだ。

さっそく，それを使ってプログラムを書いてみよう！

●図4

	回数別合計	国語	社会	理科	
科目別合計	A (0.0)	A (1.0)	A (2.0)	A (3.0)	
1回目	A (0.1)	A (1.1)	A (2.1)	A (3.1)	
2回目	A (0.2)	A (1.2)	A (2.2)	A (3.2)	
3回目	A (0.3)	A (1.3)	A (2.3)	A (3.3)	

●リスト5

```
100 DIM A(3,3)
110 FOR I=1 TO 3:FOR J=1 TO 3
120 READ A(I,J)
130 A(0,I)=A(0,I)+A(I,J)
140 A(J,0)=A(J,0)+A(I,J)
150 NEXT
160 PRINT I;"ｶｲﾒ=";A(0,I)/3
170 NEXT
200 DATA 60,70,50,30,40,80,90,100,20
300 FOR I=1 TO 3
310 READ A$
320 PRINT A$;"=";A(I,0)/3
330 NEXT
400 DATA ｺｸｺﾞ,ｼｬｶｲ,ﾘｶ
```

コンピューターは，いつも，「0」からはじまることは何回(なんかい)も言(い)ったよね。「DIM　A (3, 3)」というのは，Aという配列変数(はいれつへんすう)は，たて4×横(よこ)4，の16通(とお)りを使(つか)える，というわけだ。

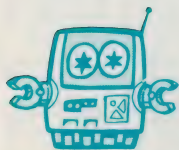
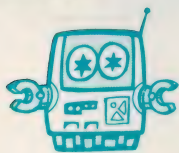
図4　と**リスト5**を見(み)くらべながら，このプログラムがどういうことをしているかは，もう，わかるよね。

005-3

文字と数字のあいだから

「ASC」や「CHR$」はもう知ってるね。文字と数字は，いろんな形でつながっている。また，コンピューターの中にはいろんな関数がある。それらをサンプルを使いながら紹介しよう。

●リスト6

```
100 A$="TESTPROGRAM"
110 B$=LEFT$(A$,4):PRINT B$
120 B$=RIGHT$(A$,7):PRINT B$
130 B$=MID$(A$,5,3):PRINT B$
200 C=LEN(A$):PRINT C
```

LEN

LEFT$

このプログラムに新しく出てきた単語は，文字列をいろいろ動かす命令なんだ。

一番最後の「LEN」というのは，文字列の長さを返してくれる。うしろの「( )」の中は，プログラムのように文字変数でもいいし，

C＝LEN（〝TESTPROGRAM〞）

のように，直接，文字列を入れてもいい。これは，他の3つの命令も同じだ。

B$＝LEFT$（〝TESTPROGRAM〞，4）

とも書けるわけだ。

110行はカッコ「( )」の中の文字列のうち，左側から4つの文字をB$とする，という命令なんだ。もちろん「,」のあとの「4」を「5」にすれば，5つの文字になることはわかるね。

120行は逆に，右側の7文字をB$にする命令だ。

A$=TESTPROGRAM

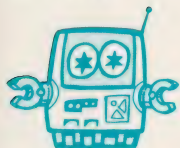
B$=LEFT$(A$,4)

C$=MID$(A$,6,2)

D$=RIGHT$(A$,3)

A$=TEST PROGRAM
B$=TEST
C$=RO
D$=RAM

130行は，文字列A$の左側から5番目の文字から，7番目の文字までの3文字を，B$に入れるという意味なんだ。

MID$

　　ＭＩＤ＄（文字列，先頭の文字は左側から数えて何番目か，いくつの文字数か）

というふうに書く。

　とすると，110行，120行はそれぞれ，

```
110  B$=MID$(A$,1,4)
120  B$=MID$(A$,5,7)
```

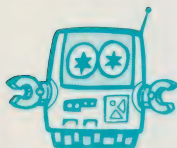
と書いても同じになるよね。これらはそれぞれ英語の，

ＲＩＧＨＴ	＝	右
ＬＥＦＴ	＝	左
ＭＩＤＤＬＥ	＝	中，中央
ＬＥＮＧＴＨ	＝	長さ

の略字なんだ。

●リスト7

```
100 A=11:B$=STR$(A):C=LEN(B$):PRINT B$,C:PRINT ASC(B$)
110 A=-11:B$=STR$(A):C=LEN(B$):PRINT B$,C:PRINT ASC(B$)
200 A$=" 11":B=VAL(A$):PRINT B
210 A$="11":B=VAL(A$):PRINT B
220 A$="-11":B=VAL(A$):PRINT B
230 A$="A11":B=VAL(A$):PRINT B
240 A$="1A1":B=VAL(A$):PRINT B
250 A$="11A":B=VAL(A$):PRINT B
```

STR$

今度は，数字を文字に，文字を数字にしたいときの命令だよ。

「STR$」というのは，うしろの「( )」の中に，数字や数値変数を入れると，そのまま文字列に直してくれる関数なんだ。

ところが，100行を実行するとわかるように，「 11」という数字を文字列にすると，その文字列の長さは「3」になってしまったね。そして，その文字列「B$」の「ASC」をとると（「ASC (X$)」とするとX$の先頭の文字のナードカンバーを教えてくれるんだったよね。）「32」になっている。これは，110行を実行してみるとわかる。

前にも言ったとおり，数を画面へ表示するとき，その数が「+」（プラス）だったら，「+」のかわりに，「 」（空白=スペース）を書くんだ。だから文字列に直したときも，そのスペースを先頭に3文字の文字列にしたわけなんだ。当然，その先頭のASCコードは空白を意味する「32」になるよね。

今まで，

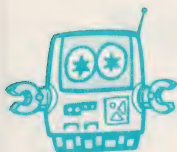
A=1：PRINT 〝A=〞；A

A= 1

となってたね。「=」のうしろにすぐ「1」が表示されなかっただろう。これを何とかしたいな，と思ったら，

A=1：A$=STR$(A)：PRINT 〝 A=〞；
LEFT$(A$，LEN(A$)−1)

とすると，ちゃんと「A=1」と表示されるんだ。

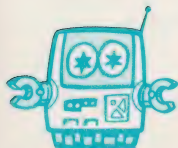
「VAL」というのは，その逆で，文字列を数値に直す命令なんだ。

文字列はスペース,「－」, 数字だけでできていなければならない。もし文字があれば，それまでに出てきた数字だけを数値に直す。230, 240, 250行は，それぞれ「Ø」,「　1」,「　11」になるだろう。(注　最初の文字が「　&H」の場合，16進数といって数値に直すことができるが，数字を16進数としてあつかう,「HEX$」と一緒にジャンプ編で説明する。)

&H
HEX$

もちろん，文字列が「"698374"」のように，あつかえる数値の制限をこえてしまうと，エラーになっちゃうよ。

今度は画面の読みとり方だ。カーソルは，X座標とY座標で決まる。いいかえれば，コンピューターの中に，いつも「X=　」,「Y=

●リスト8

```
100 CLS
110 LOCATE 5,1
120 X1=POS(0):Y1=CSRLIN
130 PRINT "TEST"
140 X2=POS(0):Y2=CSRLIN
200 A$=SCR$(5,1)
210 B$=SCR$(5,1,0)
220 C$=SCR$(5,1,1)
250 COLOR 5,1,3
260 D$=SCR$(5,1,1)
300 LOCATE0,4
310 PRINT "X1=";X1,"Y1=";Y1
320 PRINT "X2=";X2,"Y2=";Y2
330 PRINT A$,B$
340 PRINT "COLOR1=";ASC(C$)
350 PRINT "COLOR2=";ASC(D$)
```

」という数値が記憶されていて,「PRINT」命令がくると，記憶しているX, Yの位置から表示するわけなんだ。その位置を変える命令が「LOCATE」だったよね。

POS(Ø)
SCRLIN

「POS」(Ø)と「CSRLIN」は，そのXとY座標がどこにあるか

を教えてもらう命令なんだ。100行から140行で，変化をたしかめられるよ。

「POS（Ø）の「Ø」は，ダミーといって，制限以下の数字なら何でもいいんだ。あとで出てくる「MOVE」のPOSITION」という命令とは間違えないようにね！

次の200行から出てくる「SCR$」は画面のどこに，どんなキャラクターが書かれているかを教えてくれる関数なんだ。実は200行と210行は，まったく同じ働をしてるんだよ。「( )」の中の一番最後が「Ø」のときは，省略してもいい，というわけだ。この省略，あるいは「Ø」のときは，その場所にある文字を教えてくれる。逆に，最後の数字が「1」のときは，そのキャラクターのカラーを教えてくれるんだ。

SCR$

SCR$（X座標，Y座標，色＝1・文字＝Ø）

というしくみだ。

カラーは当然Ø－3の間しかなく，CHR$（Ø）やCHR$（3）は画面に出ないから，カラー情報を知りたいときは，340～350行のように「ASC」で聞かなければいけない。

当然「( )」の中の最初の2つの数値，X座標・Y座標は「6Ø,

40」のように，テレビの画面のわくをこえちゃうとエラーになってしまうんだ。

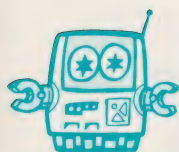
この「SCR$」は迷路ゲームなどによく使うから大事な命令だぞ。

ふつうのパソコンには三角関数（SIN，COS）や対数関数（LOG）とか，いろんな関数があるけど，ファミコンは数学の研究機械じゃ

●リスト9

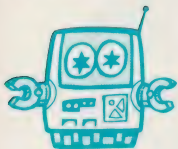
```
100 A=10:B=3:C=0:D=-10
110 E=A/B:PRINT E
120 E=A MOD B:PRINT E
130 E=ABS(A):PRINT E
140 E=ABS(C):PRINT E
150 E=ABS(D):PRINT E
160 E=SGN(A):PRINT E
170 E=SGN(C):PRINT E
180 E=SGN(D):PRINT E
```

ないので，バッサリ，カットしてあるんだ。

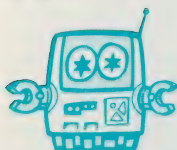
MOD

「MOD」というのは，割り算の余りを出す命令だったね。小数を使わない割り算は，

10÷3＝商 3……余り1

としたよね。この「商」を求めるのが「10／3」そして，余りを求めるのが「10　MOD　3」というわけだ。

ABS

「ABS」という関数は，うしろの「( )」の中の数値の絶対値を求める命令だ。もう学校で「｜−10｜」なんていうのは習ったかな？

絶対値というのは，「＋」とか「−」の記号とかをとった数値なんだ。この関数がないと，図5　のような時，敵とのきょりを計算するのに，

```
きょり＝X－A
```
とすると，敵とのきょりが「－4」なんて変なことになっちゃう。しかたないから，
```
きょり＝X－A：IF　きょり<0　THEN　きょり＝
    A－X
```
なんてしなきゃならない。こんなとき，
```
きょり＝ABS（X－A）
```
とすれば一発だ！

SGN

「SGN」というのは，「( )」の中の数値が「＋」か「－」か「Ø」かを教えてくれる。「＋」だったら「　1」，「Ø」だったら「Ø」，「－」だったら「－1」だ。

主人公が敵を追いかけていくゲームを作るとき，
```
IF　X>A　THEN　A＝A＋1　（右へゆく）
IF　X＝A　THEN　A＝A　　（そのまま）
IF　X<A　THEN　A＝A－1　（左へゆく）
```
とするよりも，
```
A＝A＋SGN（X－A）
```
とすると一発でOKだ！　わかるかな？

もう一つ，「A」，「B」という二つの変数の中身を交換するには
```
C＝A：A＝B：B＝C
```
のようにしなければいけないのだが，簡単に
```
SWAP　A，B
```
でできる。「SWAP」はうしろの二つの変数の値を交換しろという命令で，もちろん，文字変数同士にも使える。

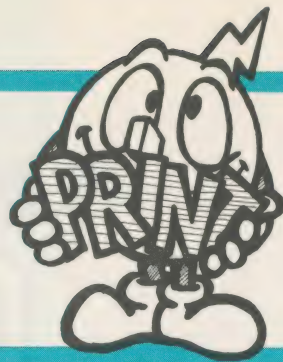
プログラムの途中でコメントを入れたい時は
```
1Ø　'TEST
2Ø　REM　MAIN
```
のように「'」や「REM」と書くと，そこからうしろの部分は命令として見えないので便利だ。

さて，もうこれで，ベーショックの基本的な命令は，ほとんどすんじゃった。意外にやさしかっただろう？　次はいよいよ，グラフィック・ワールドへ出発だ！

作戦指令006

テレビへ落書き3つの方法

006−1

PRINT文にもいろいろあるぞ

落書きなんて，とんでもない！ごめん，ごめん。

ちょっとむずかしい変数や文法なんて忘れて，これから少し，絵を書く勉強をしてみようか。

一番簡単なのは，PRINT 命令だね。ところで，「LOCATE」と「PRINT」の関係はもう頭に入ってるかな？

```
10 CLS
20 PRINT "TEST"
```

とすると，画面の左上すみに「TEST」と出るはずだね。コンピューターは「PRINT」命令に出会ったとき，どこから文字を書いたらいいか，を覚えている。その数値を知りたいときは「POS (0)」と「CSRLIN」という命令で教えてくれるはずだね。

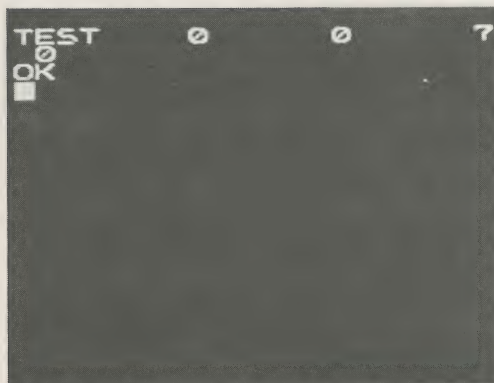
```
1Ø CLS
2Ø X1=POS(Ø):Y1=CSRLIN
3Ø PRINT "TEST"
4Ø X2=POS(Ø):Y2=CSRLIN
5Ø PRINT X1,Y1,X2,Y2
```

最初「X＝Ø，Y＝Ø」の位置にあったカーソルの位置が「TEST」と表示したあと「X＝Ø，Y＝1」になっているのがわかるよね。

ところで，50行の「，」は何なんだろうか？30行を，

```
3Ø PRINT "TEST",
```

としてみよう。

今度は「TEST」と書かれた同じ行に，

X1，Y1，X2，が表示されて，しかもY2の値は同じで，X2の値だけが「7」に変化しているね。

なぜかというと，「PRINT」命令で表示させる文字や数字の一番うしろに，何にもないときは，カーソルの位置は次の行の一番左（X2＝Ø，Y2＝Y1＋1）にうつる。それに対して「，」が最後に書かれているときは，Y2が変化しないで，X2が変化するんだ。

ファミコンのキャラクター画面は，横28字（X＝Ø～27）だったね。「PRINT」文の最後に「，」か「；」がないときは，「改行」といって，次の「PRINT」命令で何かを表示するときは，次の行から書きはじめなさい，という意味になるんだ。

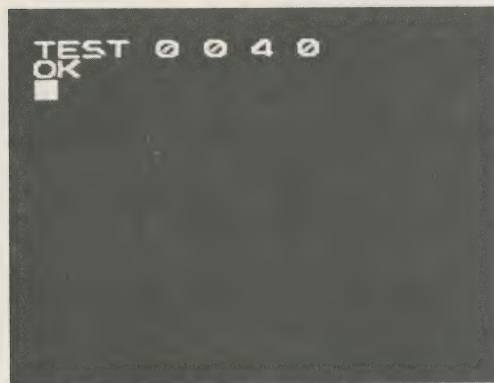
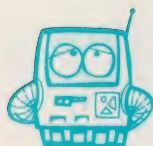
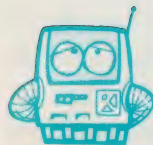
```
3Ø PRINT "TEST";
5Ø PRINT X1;Y1;
        X2;Y2
```

としてみよう。今度は同じ行につづけて，

```
TEST Ø Ø 4 Ø
```

と表示される。それぞれの数字の前に一字分空白（スペース）があるのは，数値を表示する時の約束で「＋」の意味だったよね。

;

,

つまり，「；」があるときは，カーソルの位置を変えない（改行しない）で，続けて書け，という意味なんだ。

「，」があると，X2 は，

```
X2=X1+(A/7+1)*7
```

直前に表示されるものが文字列（A$）のときは，

```
A=LEN(A$)
```

数字（X）のときは，

```
A=LEN(STR$(X))
(=その数字(X)のケタ数+1)
```

であらわされる。つまり，横28文字を7ずつの4つのブロックに分けて，その前に表示されたものと最低1つの空白をあけて表示しろ，ということになるんだ。

```
3Ø PRINT "7WORDS7",
3Ø PRINT "6WORDS",
```

など，30行の「PRINT」文のあとの文字列の長さをいろいろかえてみると，働きがよくわかるだろう。

```
1Ø CLS
2Ø FOR I=Ø TO 15
3Ø FOR J=Ø TO 15
```

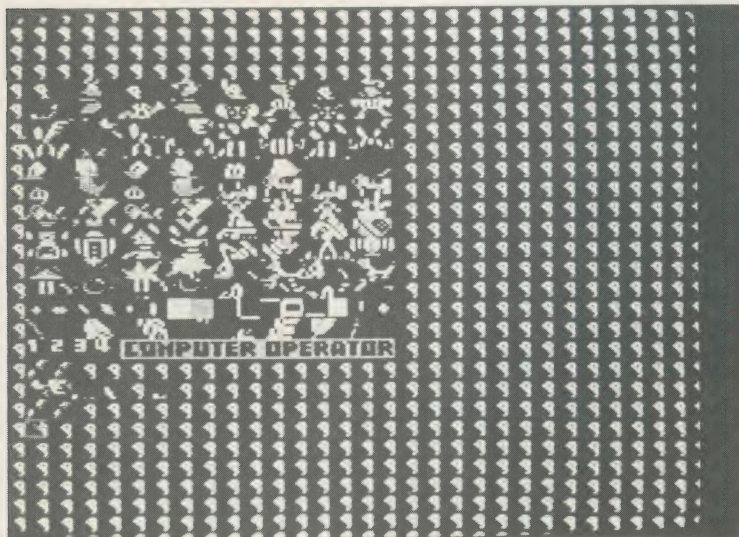
```
40 LOCATE J,I:PRINT CHR$(I*
        6+J);
50 NEXT:NEXT
```

このプログラムを打ち込でRUNしながら，**キャラクタコード表B**を見てみよう。CHR$ (Ø) からCHR$ (31) まで (キャラクターテーブルBのA0からD7の32個)が表示されない他は，コード表Bの通りに表示されるね。(CHR$ (32) は空白＝スペースだ)

コントロール・コード

この0から31は普通，**コントロール・コード**といってコンピューターがいろいろな操作をやるときに使っているんだけど，ファミコンではできるだけ多くのキャラクターを表示したいため，この場所にもキャラクターを割り当てて2重に使っているんだ。まえの「INKEY$」でやったように，たとえばカーソルキーの「▼」のコードナンバーが「31」だったね。だからBG画面ではこの0から255までのキャラクターが全部使えたけど，普通のベーシック画面で「PRINT　CHR$ ( )」として表示されるのは，32から255まで，となってるんだ。

じゃあ，**キャラクターテーブルA**のキャラクターは使えないのか，というとそうでもない。

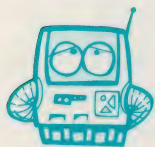
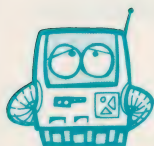
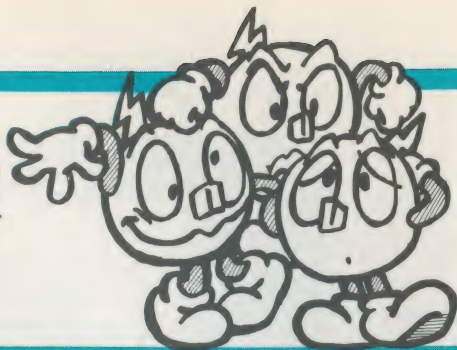
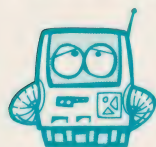
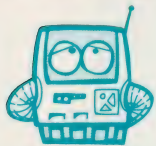
前のプログラムを「RUN」して，各キャラクターが表示された所

CGEN

で，ダイレクトに「CGEN　0」としてごらん。画面がむちゃくちゃになったね。でもよく見ると，テーブルBのキャラクターが書れていた所に，マリオやレディが書かれているよね。

画面全体に書かれている模様はなんだろう？

スペース（空白）も一文字で，コードナンバーは「32」だってのは何回も言ったよね。ところで，テーブルAのコードナンバー「32」はなんだろう？レディの横顔だね。画面一杯にレディの横顔が「スペース」のかわりに表示されているんだ。いくらレディがかわいいといっても，ちょっと気持ち悪いなぁ。

Ctr—D

このままじゃ，なんにもできないから「コントロールキー」を押しながら「D」のキーを押してごらん。もとにもどったね。(Ctr—Dというように書いてあると，コントロールキーを押しながらDのキーを押す，という意味だよ。)

006−2 ファミコンは何重人格？

「CGEN」とはどんな命令だったのか？その前に，ファミコンの顔（画面）は，一体どうなっているんだろう？

アニメ映画を作るとき，普通，「セル」と呼ばれる透明なプラスチックにいろんな絵を書いて，重ね合わせていることは君達も知ってるよね。（図1）

Ⓐ，Ⓑ，Ⓒの重ね合わせで，太陽が山のうしろにかくれたり，汽車の下にある線路が見えなかったりするわけだ。

バックドロップ

ファミコンも考え方は同じなんだ。まず，一番下に「**バックドロップ**」面がある。この画面に文字やキャラクターは書けなくて，何か一色にぬる事ができるわけだ。ベーシック画面ではいつも黒になってるね。背景画面とでも呼ぼう。そして「**キャラクタ画面**」，ファミコンのマニュアルでは「バックグラウンド面」になってるけど，意味からするとこっちの名前が合ってると思うね。つまり文字・数字・キャラクタテーブルＢなどを表示する面で，ベーシックのプログラムを使ったりするのもこの面だ。それに「CGEN」命令を使うと，キャラクタテーブルＡも使えそうだぞ。

キャラクタ画面

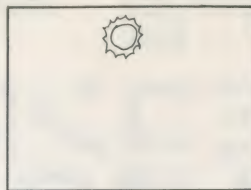
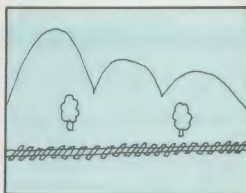
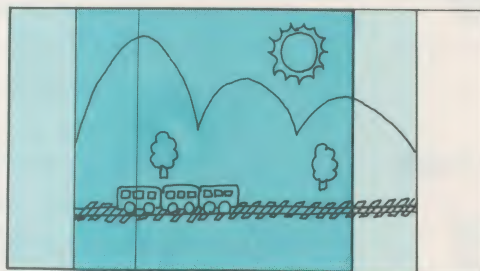
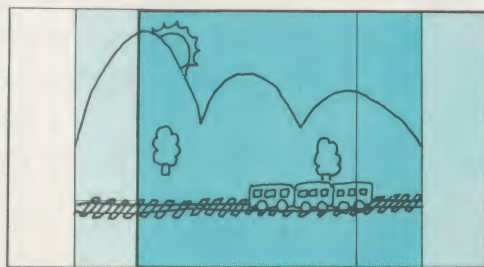
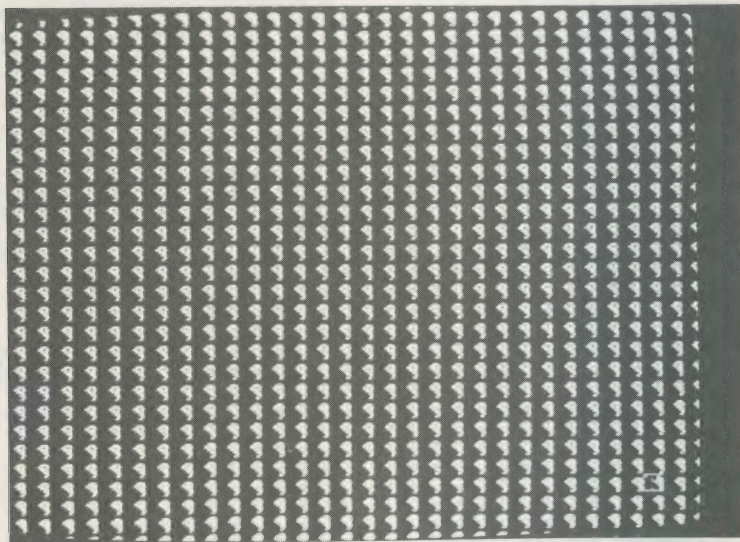
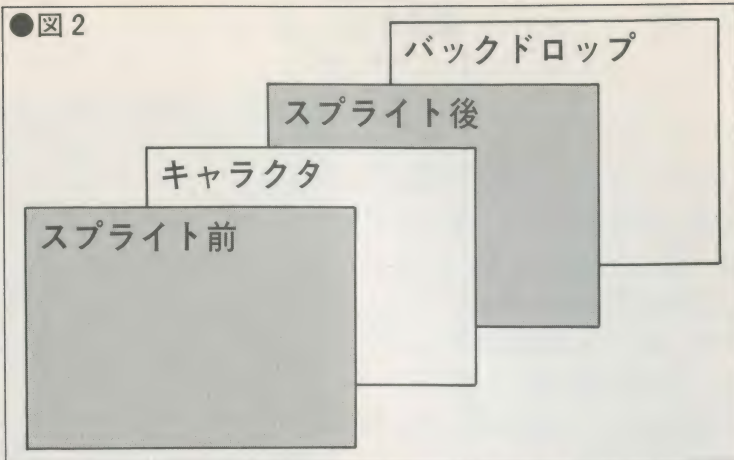
スプライト面

そして問題の「**スプライト面**」だけど，このスプライト面は，2枚あるんだ。おもに，キャラクタテーブルＡのキャラクターを表示したり，動かしたりする。この2枚はそれぞれ，「スプライト前」と「スプライト後」ともいえるんだ。

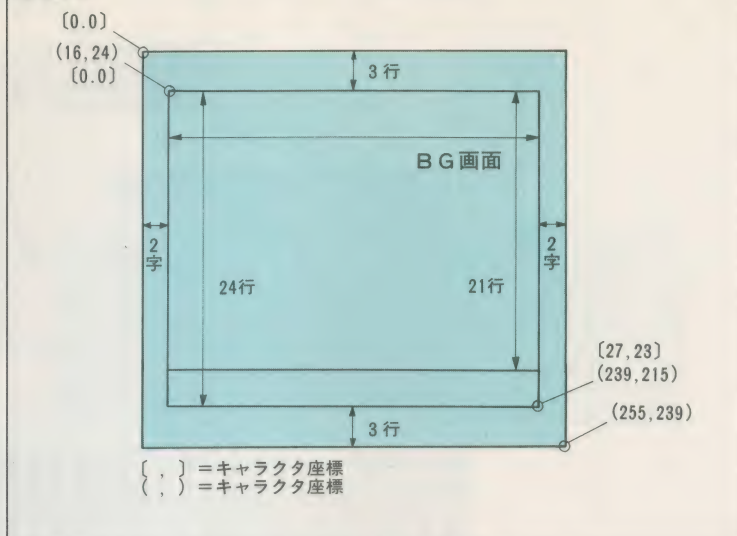
つまり，一番下に「バックドロップ」面があり，次に「スプライト後」，「キャラクター」，一番上が「スプライト前」になってるんだ。（図2）

この4枚のうち，1枚だけ使える大きさが違うんだ。そして，「CLS」をしたあと，「CGEN　0」をもう一度してみよう。

●図1
A
B
C
C B A
A B C

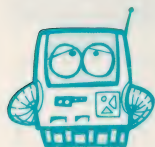
●図2

レディの横顔オンパレードだけど，カーソルは何とか見えるね。カーソルキーでカーソルを移動させてみよう。画面にはレディが出てるのに，移動できない所があるだろう。移動できる所を数えてみると，ちゃんと28×24あるはずだね。(Ctr—Dで元に戻そう。)

ファミコンの画面は，本当は横32文字，たて30行もあるんだ。ただ，家庭用のテレビは，いろんな会社のがあって，全部をきれいにうつせないかもしれないから，キャラクター面は，そのうち28×

●図3

24（図3）しか使ってないんだ。上下左右に余った分は，スペース（空白）でうめてるのは，さっきの実験でわかるよね。

BG画面はどうなんだろう。ホップ編で練習したように，BG画面でいろんな絵を書いたあと，ベーシック画面に来たら，せっかく書いた絵が消えてしまった。と思ったら「SYSTEM」命令でBG画面に行くとちゃんと残っていたね。

今，ためしに，BG画面でいろんな絵を書いてみよう。(かんたんな絵でいいんだ。)そのあとベーシックにもどってみると，まっくらになってるね。ダイレクトで，

VIEW

```
VIEW
```

と打つと，今かいた絵が出てきたね。

```
CLS
```

とするか，シフトキーを押しながら，CLR キーを押すと，当然，今の絵は消えてしまう。もう一度，

VIEW

とすると，また元の絵が出てくるだろう。

つまり，BG 画面でのお絵かきは，画面に表示する「キャラクター」面とは，全然別の所で行なっている。そして出来上がった絵は「VIEW」という命令で，そっくりそのまま「キャラクター」面へ移動(転送という。)されたわけだ。だから「キャラクター」面を「CLS」命令で消してしまっても，もとの所にはその絵が残っているんだから，もう一度「VIEW」命令で「キャラクター」面に同じ絵が出てくるしくみなんだ。(「VIEW」命令は，BG 画面を使うゲームには絶対必要な命令なのだ！)

この BG 画面は横28字，たて21行で，キャラクター面の下３行を残した上の部分へ，そっくりそのまま転送される。

ところで，文字や，グラフィックスのキャラクターは，すべて「８×８」の小さな点によって作られている。というのは覚えてるかな？（ホップ編85ページ）

おもに，マリオやレディなどのグラフィックキャラクターを動かすのに使うスプライト面は，その動きを細かく表現するために，画面に表示する単位を，この小さな点（ドットと呼んだよね。）で示している。このスプライト面の大きさは，バックドロップ面と同じ32×30行だ。１字をドットで表わすと８×８だから，スプライト面をドットで表わすと，

X…８×32＝256ドット　X座標…Ø～255

Y…８×3Ø＝240ドット　Y座標…Ø～239

ということになるよね。

```
1Ø  SPRITE ON
2Ø  DEF SPRITE Ø,(Ø,1,Ø,Ø,Ø
 Ø  )="DEFG"
3Ø  INPUT "X=",X
4Ø  INPUT "Y=",Y
5Ø  SPRITE Ø,X,Y
6Ø  GOTO 3Ø
```

```
OK
10 SPRITE ON
20 DEF SPRITE 0,(0,1,0,0,0)=
"DEFG"
30 INPUT "X=";X
40 INPUT "Y=";Y
50 SPRITE 0,X,Y
60 GOTO 30
RUN
X=232
Y=208
X=
```

これはアキレスをINPUTした座標に表示するプログラムなんだ。X, Yとも0～255の間の数字ならエラーにならず, X＝0～240, Y＝5～220以外の数値だと，どこかがちょん切れるだろう。またX＝16, Y＝24でキャラクター面の左上すみ, X＝232, Y＝208で右下すみのところに，アキレスの左上がくるのがたしかめられるかな？図3を見ながら，いろいろとためしてみよう。

0～255のドット座標系と「LOCATE」などにつかう0～27の座標系と2つも出てきて，頭がおかしくなっちゃうかもしれないけど，2つの区別は，必ずつけられるようにしようね。

この2つの関係は，

X1，Y1＝キャラクター面のX，Y座標
X，Y＝スプライトのX，Y座標

とすると，

X＝（X1＊8）＋16
Y＝（Y1＊8）＋24

になっている。

006−3
スプライトが重なると

ホップ編で，スプライトが横に4つ以上重なると表示されない，ということにふれたけど，どういう事なんだろう？

```
100 SPRITE ON
110 FOR I=0 TO 4
120 DEF SPRITE I,(0,1,0,0,0
    0 )="DEFG"
130 SPRITE I,100,I*20+10
140 NEXT
```

アキレスが，たてに5つ表示されたね。今度は，

```
130 SPRITE I,I*30,100
```

にしてみよう。4つしか出てこないね。

130 SPRITE I，I＊30，100＋I＊2

とすると，少しずつ高さがずれながら4つのアキレスが表示されたあと，頭の欠けた5つ目のアキレスが出てきたね。

キャラクターの所で，ファミコンはセルを重ねて書いているみたいなもんだ，ということを言ったよね。ところで，このスプライト

面にはセルを使う制限があるんだ。たとえば，アキレスやマリオは，たて16，横16のドットで表わされてることは前にも言ったね。このドット１つが１cm×１cmもある大きなものとしよう。

図４のⒶのような，横に長く，厚さも１cmあるすごく厚いセル(棒みたいなもんだ。)のまん中あたりに16cmのキャラクターがある。つまり「DEF　SPRITE」なんかで，決まったキャラクターは，このⒶがたて16本ならんだものと考えられるね。とすると，プログラム①の場合は，図４—Ⓒのようになる。それに対し，②にすると，Ⓔのようになるのがわかるね。横から見たのが，それぞれⒹとⒻだ。

ではプログラム③はどうだろう。図４—Ⓖになり，横から見たら５Ⓗになるのがわかるかな。

このスプライト面に許されているセルの厚みは４cm，つまり４枚だけなんだ。そこでプログラム②，図４—Ⓕでは４枚目までしか表示されなくて，③，図４—Ⓗでは５枚目（１番上）の絵は頭（図では左）の①の部分が厚さ５cmで表示されないけど，残りは表示された，というわけなんだ。

４枚目まで

スプライトを使ってゲームを作るときは，この点に注意しなきゃいけないよ。

ところで，さっきから，

```
DEF SPRITE ～＝"DEFG"
```

なんて使ってきたけど，これはどういうことなんだろう？

アキレスを表示するには，たとえば，

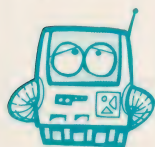
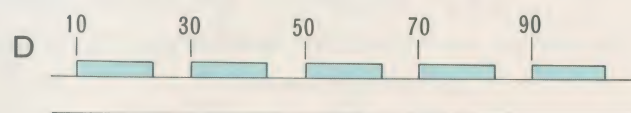
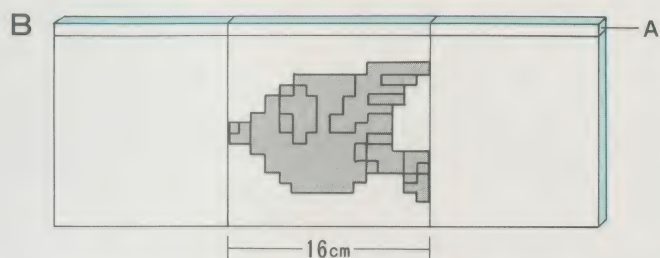
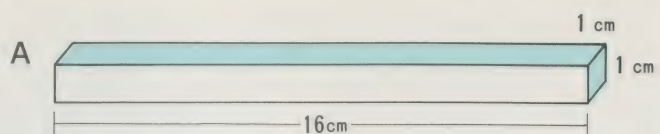
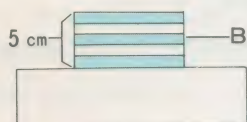
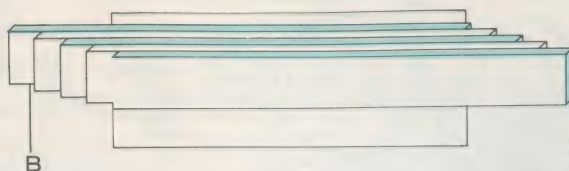
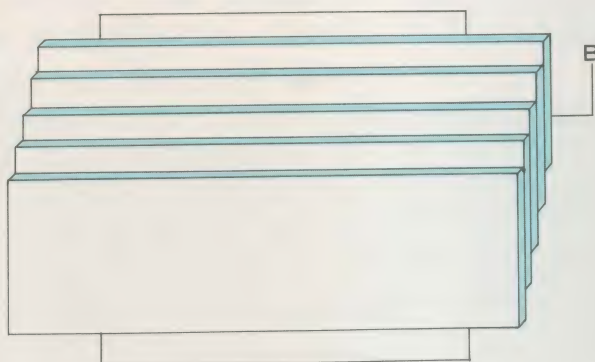
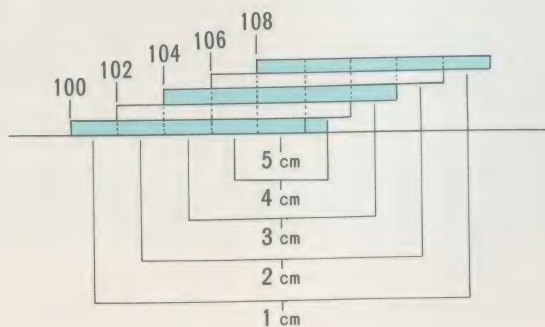
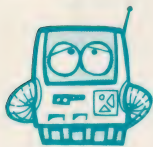
```
DEF SPRITE～＝CHR$(64)+CHR$(
65)+CHR$(66)+CHR$(67)
```

のはずだったよね。ところで，CHR$（64）から（67）は，それぞれ，キャラクタテーブルBでは〝D″から〝G″だったね。そこでメモリーや入　力をやさしくするために，使ったわけだ。

このように，キャラクタテーブルAとBとは，コードナンバーは同じでも，その数値がどちらのテーブルを使うかさえ決めればいいわけなんだ。

電源を入れた状 態では，キャラクタ面にはテーブルB，スプライト面にはテーブルAを使うと決まっているんだけど，それを君達が

●図4
A
1 cm
1 cm
16cm
B
A
16cm
C
10
30
50
70
90
B
D
10
30
50
70
90

●図 4
E
B
5 cm
B
F
B
G
100
102
104
106
108
5 cm
4 cm
3 cm
2 cm
1 cm

かえることもできるんだよ。

CGEN

「CGEN」という命令がそれなんだ。(やっと出てきたね。)

	キャラクター面	スプライト面
CGEN Ø	テーブルA	テーブルA
CGEN 1	テーブルA	テーブルB
CGEN 2	テーブルB	テーブルA
CGEN 3	テーブルB	テーブルB

というのが「CGEN」と，そのあとに続く数値の意味なんだ。

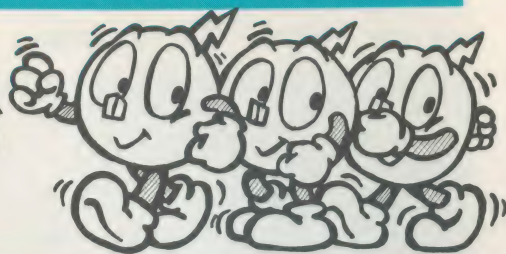
電源を入れた直後は，「CGEN　2」になってるわけだね。そして「Ctr—D」というのは，「CGEN　2」の命令だったとも言えるよね。

この「Ctr—D」。他にもいろんな働きを一度にしてくれるんだ。

「SPRITE　ON」でスプライト面を一度表示すると，「CLS」命令では，スプライト面の表示は消えなかったね。「CLS」は，キャラクタ面を消すだけの働きをするからなんだ。スプライト面を消すには，「SPRITE　OFF」という命令しかなかったね。

ところが，この「Ctr—D」，さっきの「CGEN　2」とこの「SPRITE　OFF」，そしてジャンプ編で習う，色の組み合わせを変えたものを元にもどす，という働きを一度に行なってしまうんだ。

「Ctr—」，つまり，「CTR」キーを押しながら，他のキーを押す，というのには，いろんな種類があって，とても便利なんだ。くわしい説明は巻末の作戦資料103を見てほしい。

006−4
スプライトを動かすには

ホップ編では，「スプライト命令」を使って，スターシップとスターキラーを動かしたよね。ところが，キャラクタテーブルAを見てみよう。マリオだけで7種類もあるね。「ドンキーコング」や「マリオブラザース」なんかを見てると，マリオがチョコチョコと手足を動かしながら歩いている。この感じを出すには，キャラクタテーブルAのマリオの「WALK1」，「WALK2」，「WALK3」を順番に少しずつ動かしながら表示しなければいけないんだ。

●リスト1

```
100 CLS
110 SPRITE ON
120 FOR I=0 TO 2
130 DEF SPRITE I,(0,1,0,0,0)=CHR$(I*4)
+CHR$(I*4+1)+CHR$(I*4+2)+CHR$(I*4+3)
140 NEXT
150 FOR I=200 TO 0 STEP -1
160 SPRITE I MOD 3
170 SPRITE (I+1) MOD 3
180 SPRITE (I+2) MOD 3,I,100
190 PAUSE 3
200 NEXT
```

これが，マリオをX=200，Y=100の位置から移動するプログラムなんだ。120行から140行でマリオの歩く3つのポーズを，それぞれ「SPRITE」の0から2に定義しているのがわかるね。

150行から200行でマリオを移動している。

Iの値	消すスプライト	描くスプライト
200	2，(0)	1
199	1，(2)	0
198	0，(1)	2
197	2，(0)	1
196	1，(2)	0
∫	∫	∫

160行　170行　180行

3つのスプライトが重ならないように，まず他の2つを消したあと，使うスプライトを描くために「MOD」を使っている。(170行

●リスト2

```
100 CLS
110 SPRITE ON
120 FOR I=200 TO 0 STEP -1
130 J=I MOD 3
140 DEF SPRITE 0,(0,1,0,0,0)=CHR$(J*
4)+CHR$(J*4+1)+CHR$(J*4+2)+CHR$(J*4+3)
150 SPRITE 0,I,100
160 PAUSE 3
170 NEXT
```

は，なくてもよいが，わかりやすくするために，付け加えた。)

また，次の方法もある。

これは「SPRITE　0」しか使わずに，その中身を，そのたびに書きかえる。という方法だ。

ところが，どちらの方法を使っても問題は残る。マリオは必ずしも左向きだけ，とは限らないんだ。ホップ編を思い出そう。マリオを右向きにするには，「CHR$」の並べ方を変えた上に，「DEF SPRITE」の「( )」の中に4番目の数字（4番目のパラメーターと

いう。)を1にしなきゃいけない。その上，いろんなキャラクターを表示していると，メモリーがたりないし，第一，「DEF　SPRITE」

●リスト3

```
100 CLS
110 SPRITE ON
120 DEF MOVE(0)=SPRITE(0,7,1,1,0)
130 POSITION 0,200,100
140 FOR I=0 TO 100
150 MOVE 0
160 IF MOVE(0)=-1 THEN 160
170 NEXT
```

は0から7までの8つしか定義できない。

そこで，新しい命令だ。

新しい命令が，いくつか出てきたね。

DEF MOVE( )=SPRITE( )
MOVE
MOVE( )
POSITION

そして，あとで出てくる，

CUT, ERA, XPOS, YPOS

と，Ver.3で使える，

CAN, CRASH, VCT

MOVE系

をふくめて，「MOVE=ムーブ」系のコマンド（命令）群とよぼう。

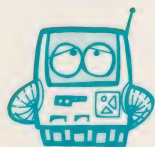
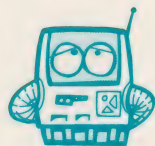
それに対し，今まで使ってきた，

DEF SPRITE n, (　　　)=文字列
SPRITE n, X, Y
SPRITE ON
SPRITE OFF

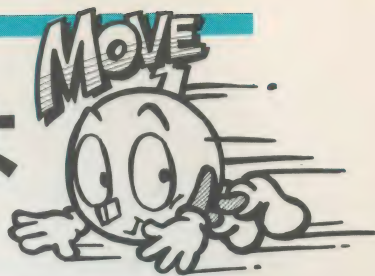
は，「SPRITE」系の命令群といえる。

スプライト系

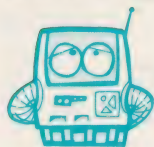
つまり，「スプライト」面にキャラクターを表示するには「MOVE」系と「SPRITE」系の2つの流れがあるわけだ。

この「MOVE」系の命令は「SPRITE」系の命令をより進めたも

ので，他のパソコンにはない強力な命令になっているんだ。
君達も，どこかの雑誌で読んだかもしれないけど，一つのキャラクターを描いて，それを移動させるためには，かなり多くのプログラムとメモリーを使うんだ。ところがファミコンには，わずか1982バイトしかメモリーがない。そこで普通のパソコンのように絵を書くための命令は，全部なくして，ファミコン内部に用意してあるキャラクター(テーブルA，B)を自由に使えるようにした，というわけなんだ。
ファミコンとパソコンで一番違うのは，この「MOVE」系の命令群にある，って言っても間違いじゃないんだ。
その他では，パソコンの持っている機能をけずって小さくしているファミコンだけど，これだけは，兄貴分のパソコンよりもずっと進んでいるんだ。(何しろ「SPRITE」系の機能でさえ，MSXやX－1など，一部のパソコンしか持ってないんだから。)

006−5

MOVEというのは動くという意味

DEF MOVE
MOVE

さて，いよいよ「MOVE」の脱明だよ。

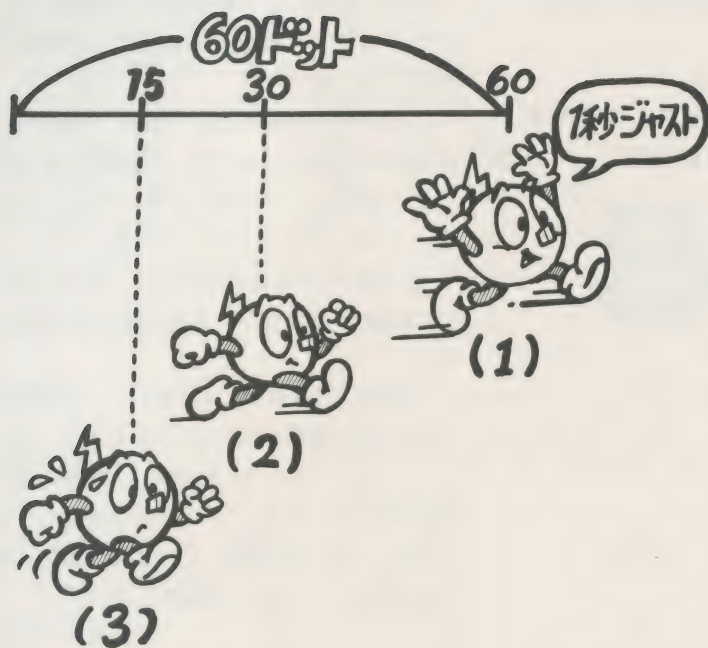
DEF MOVE (n)=SPRITE (A, B, C, D, E, F)

この命令系の特徴は，「DEF　MOVE」であるキャラクターとその動きを定義しておくと，「MOVE」という命令で自動的に定義された動きを行なってくれる，という点だ。しかも，「DEF　SPRITE」のように，キャラクターが右向きとか，ななめとかで，その並び方を変えたりする必要がなく，キャラクターの名前だけで，どの「CHR$」をどういうふうに組みあわすかを自動的に決めてくれるんだ。

「DEF　MOVE」も「DEF　SPRITE」のように，0から7までの8個を登録できる。そのナンバーが，最初の「( )」の中に入る。つまり，「n」は0から7の間の数字が入るんだ。(DEF SPRITEは，このナンバーだけが逆に「( )」の外だったね。)

次に「＝」の右側の「( )」の中だ。「A」には0～15の数が入る。これはキャラクターの種類だよ。

0＝マリオ	8＝スターキラー
1＝レディ	9＝スターシップ
2＝ファイターフライ	10＝爆発
3＝アキレス	11＝ニタニタ
4＝ペンペン	12＝レーザー
5＝ファイアーボール	13＝シェルクリーパー（カメ）
6＝車	14＝サイドステッパー（カニ）
7＝スピナー	15＝ニットピッカー（トリ）

「B」は動きの方向だ。

```
8   1   2
 ↖ ↑ ↗
7 ← Ø → 3      Ø ～ 8 が入る
 ↙ ↓ ↘
6   5   4      Ø で停止
```

「C」は移動の速さだ。60ドットを移動するのに，何秒かけるかを表わすんだ。1～255までの数字を入れる。1だと60ドットを1秒という意味だ。(Øを入れると256を入れたと同じ，つまり一番遅くなる。)

「D」は「MOVE」命令を一回出したとき，何ドット移動するか，移動量を入れるんだ。1～255の数値を入れる。入れた数値の2倍のドットを移動するから，Øを入れると表示されないよ。

「E」はØか1を入れる。Øだと，このキャラクターは「スプライト前」面，1だと「スプライト後」面に書かれるんだ。

「F」はカラーのパレットナンバーで，Ø～3の数字が入るんだ。

「E」と「F」は省略して，

```
DEF MOVE(Ø)=SPRITE(Ø,1,1,5)
```

のように書くこともできるんだ。

POSITION

こうやって定義したキャラクターを，一番最初どこに表示するか，スタート地点を決める命令が「POSITION」命令なんだ。

```
POSITION Ø,1ØØ,5Ø
```

というのは，「DEF　MOVE」命令で定義したナンバー「Ø」のキャラクターは「X＝1ØØ，Y＝5Ø」の地点からスタートしたよ。という意味なんだ。

MOVE

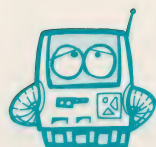
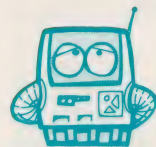
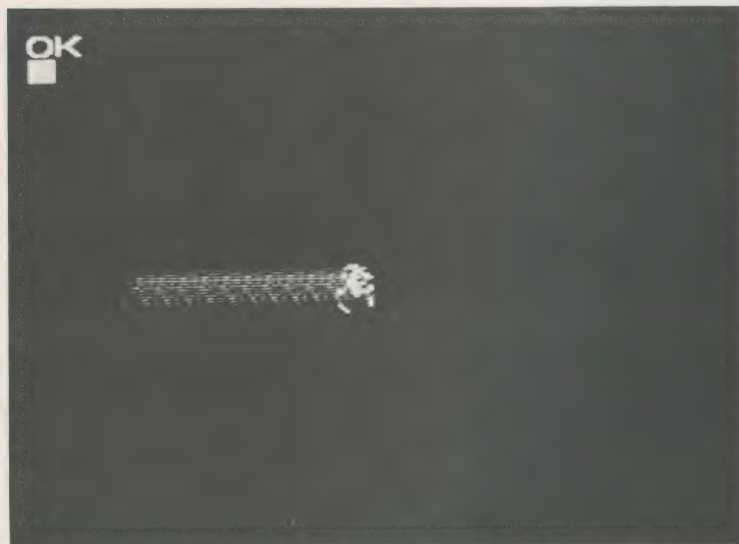
さあ，これですべてを定義した。いよいよ動かすぞ！

```
MOVE Ø
```

で，このキャラクターは，最初に定義した動きを一回だけ行なう。

```
DEF MOVE(Ø)=SPRITE(Ø,1,2,1Ø)
```

と定義していれば，1秒間3Øドット（6Øドットを2秒だから）の速度で1Øドットだけ（つまり1/3秒間）上へ移動するわけだ。

この「MOVE」，動かしたりキャラクターがいくつもあるときは，

```
MOVE Ø,1,4,5
```

のように，一度につづけて書くこともできるんだ。

```
10 SPRITE ON:CLS
20 DEF MOVE(Ø)=SPRITE(Ø,3,3
      ,255)
30 POSITION Ø,Ø,1ØØ
40 MOVE Ø
```

RUNすると「OK」の表示が出ているのに，マリオがどんどん走っている。「LIST」をとったり，コンピューターに他の仕事をやらしても，マリオは動いている。マリオは画面を2回横切ってやっと停止した。

これは，20行でマリオの1回の動きで「255」つまり2倍の51Øドット移動すると決めていたので，「動け＝MOVE」と命令したら，51Øドット移動し終るまでマリオは意地になって（？）動いているんだ。ところがコンピューターの方は「MOVE」って命令を出したら，もう自分の仕事はすんだから，他の仕事ができるようになったんだ。

あやつり人形の使い手が8人いて，コンピューターは舞台監督。「DEF　MOVE」で人形使いの一人一人にどの人形を，どういう風に動かすかを指令していると思えばいいね。「ハイ，Ø番さん」というと，Ø番の人形使いは約束通り，マリオ人形を動かしているわ

けだね。

```
6Ø MOVE Ø
```

を追加しよう。

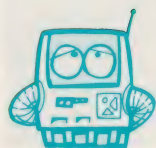
「MOVE」命令一回で，マリオは画面を2回横切ったから，命令を2回にしたら，4回横切るだろう？

残念でした！この人形使いは，仕事熱心だから，1回命令をもらって実行中（マリオを移動させている間）は，次の命令が聞こえなくなってるんだ。だから，マリオは2回しか画面を横切らない。

どうしても4回横切らすには，1回目の「MOVE」命令をやり終えるまで時間を待ったあと，2回目の「MOVE」命令を出さなきゃいけない。時間待ちは「PAUSE」だったから，

MOVE（　）

```
5Ø PAUSE 25ØØ
```

ところが，今度は，1回目の命令のあと，しばらく止まっている。「PAUSE」のあと数値が長すぎたんだ。そこで登場するのが「MOVE（ ）」だ。

```
5Ø IF MOVE(Ø)<>Ø THEN 5Ø
```

これは，もし「MOVE」命令で動きはじめたキャラクターがまだ動いていたら，50行をもう一度，つまりこのまま待っていて，動き終えたら次の行へ行きなさいよ，という命令になってるんだ。

この「( )」の中の数字は「DEF　MOVE」や「MOVE」で指定したキャラクターナンバーだ。

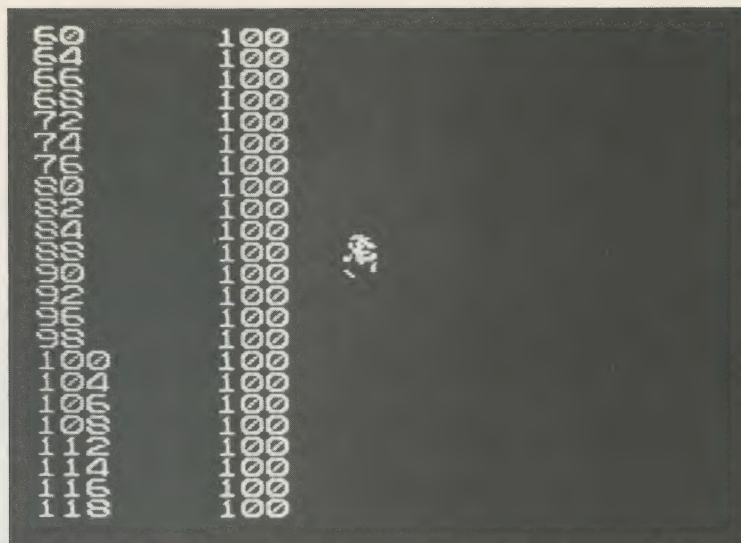
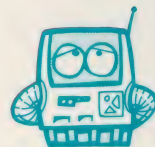
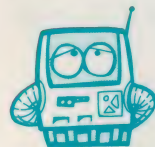
```
5Ø PRINT MOVE(Ø)
55 GOTO 5Ø
```

と直して「RUN」してみよう。キャラクターが動いている間は，ずっと「－1」で，動き終えて止まったとたんに，「Ø」になるのがわかるね。つまり，「MOVE( )」は，そのナンバーのキャラクターが移動しているかどうかを知る関数だったんだ。

55行を消して，もう一度，

```
5Ø IF MOVE(Ø)<>Ø THEN5Ø
6Ø MOVE Ø
```

としてみよう。

今度はちゃんと続けて4回マリオが画面を移動したね。1回目の命令を終えるまで待ってから，2回目の命令を伝えたんだから，人

XPOS (　)
YPOS (　)

形使いナンバー「0」も，今度は聞こえたわけだね。

```
50 PRINT XPOS(0),YPOS(0)
60 GOTO 50
```

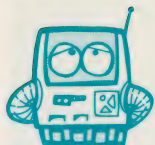
としよう。画面には次々と数字が出てくるね。

この「XPOS ( )」と「YPOS ( )」は，それぞれ「( )」の中のナンバーのキャラクターの，現在のX座標，Y座標を教えてくれる関数なんだ。カーソルキーのX座標を教えてくれる「POS」と違って，　～255のどれかの数字が出てくる，というのはわかるよね。

じゃあ，これを使ってキャラクターが，ある場所へきたら止めることはできるんだろうか？

```
50 IF XPOS(0)>100 THEN CUT
      0:GOTO 70
60 GOTO 50
70 PAUSE
```

CUT

これで，OKだ！マリオが画面まん中近くで止まっただろう。「CUT」というのは，動いているキャラクターを，その場所で停止させる命令なんだ。XPOS (0) が100をこえると停止させて70行へいく。70行では「PAUSE」でうしろに数字がないから，何かキー入力があるまでは，待っている，ということなんだ。

ここで何かキーを押すと，またマリオは動き出し，残っていた画面横断，一回半をやりとげる。

50行の「CUT　0」を「ERA　0」に変えてみよう。

ERA

今度は，マリオが消えちゃった。キーを押してもプログラム終了の「OK」が出ただけで，マリオは消えたままだ。この「EAR」は，動いているものも，止まっているものも，全部消しちゃう働きをするんだ。

ダイレクトに，

MOVE Ø

を実行してみよう。さっき消えた所から，マリオが出てきて，やはりさっき中断した動作の残り，一回半，画面横断をやっただろう。

ここで注意しなくちゃいけないのは，「CUT」にしろ，「ERA」にしろ，あくまで最初の命令の動作を中止しただけで，そのあとの「MOVE」命令は，残りの部分を再開しろということにしかならないんだ。

マリオが，画面中央まできたら，一度停止したあと何かのキーを押したら，新しく動作を始める（画面中央から2回画面を横切る）ためには，

●リスト4

```
10 CLS:SPRITE ON
20 DEF MOVE(0)=SPRITE(0,3,3,255)
30 POSITION 0,0,100
40 MOVE 0
50 IF XPOS(0)>100 THEN 100
60 GOTO 50
100 CUT 0
110 BEEP:PAUSE
200 MOVE 0
```

ではだめで，

●リスト5

```
10 CLS:SPRITE ON
20 DEF MOVE(0)=SPRITE(0,3,3,255)
30 POSITION 0,0,100
40 MOVE 0
50 IF XPOS(0)>100 THEN 100
60 GOTO 50
100 CUT 0
110 BEEP:PAUSE
200 DEF MOVE(0)=SPRITE(0,3,3,255)
210 MOVE 0
```

のように「DEF　MOVE」で再定義しなけりゃいけないんだ。この場合は「POSITION」命令でスタート位置を指定しなくても，前のX，Y座標が残っているから，元の位置から動きはじめるんだ。一度定義したキャラクターナンバーのX，Y座標は，「POSITION」命令（Ver.3の「CAN」か）で指定しない限り，いつまでも残っているんだ。

「DET　MOVE」でキャラクターを動かす人形使いさん，ガンコでゆうづうが聞かない分，記憶力はバツグンだね！

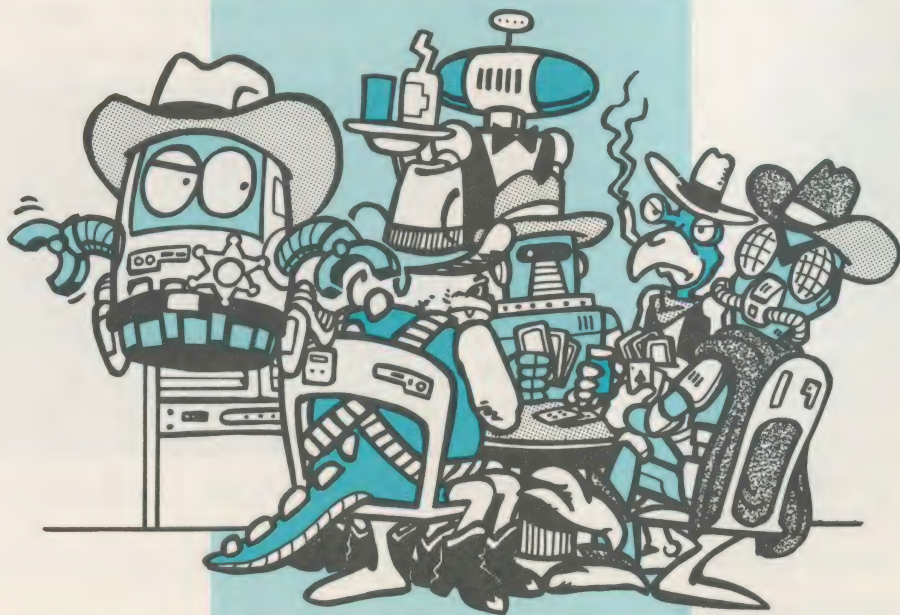
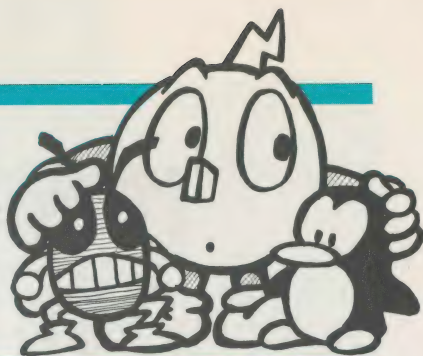
ところで，この「CUT」，「ERA」ともに「MOVE」と同じく，

CUT Ø，1，5

のように，いくつものキャラクターに一度に命令が出せるのも覚えておこう。

さて，いよいよ，ゲーム作りに挑戦だ！

作戦指令007

おっかけゲームに挑戦だ

007−1 どんなゲームを作ろうか

前に作ったゲームは敵との撃ち合いだったね。今回はどんなゲームにしようか？

BG画面を使って，道を書いて，主人公が逃げまわる。敵はファイターフライにしよう。ファイターフライが，主人公ペンペンを追いかけまわすことにしよう。

ただ逃げるだけじゃつまんない。歩いていった道にはマークをつけて，全部の道を歩き終わったら，その面は終わり，というようにしよう。

一面を終わると次の面に進むんだけど，そのたびにBG画面のデーターをLOADするのも時間がかかるね。

それじゃあ，一面終えて，レベルが上がるたびに，1個ずつカベをふやして通れない所を多くすると，むずかしくなるわけだ。しかも，ふえた分のカベは，ペンペンは通れないけど，ファイターフライは通れてしまうんだ。

まず元になる迷路をかいてみよう。

BG画面へ

SYSTEM　　(Ver.3なら**BGTOOL**)

でBG画面を呼び出そう。使うキャラクターはテーブルBの「I6」,「I7」,「J0」,「J0」,「J1」,「J2」,「J3」,「J4」,「J5」,「J6」,「J7」だ。コード表Bでいうと「CHR$ (222)」から「CHR$ (231)」までを使って書いてみる。画面をこしたかどうかの判定も一諸にやりたいので，まわりもすべて，このキャラクターを使って囲んでしまう。

とりあえず，できたのが図1だ。もちろん，この面は自分でいろいろ作ってみるのもいいだろう。

今度のゲームは，いろんな面を作っても楽しめるのでテープへの

●図1

	Ø					5					1Ø					15					2Ø					25		27
Ø	I6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	I7Ø
	J2Ø																											J2Ø
	J2Ø																											J2Ø
	J2Ø			J6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J7Ø					J4Ø											J4Ø			J2Ø
	J2Ø													J2Ø					J6Ø	J3Ø	J3Ø	J7Ø			J2Ø			J2Ø
5	J2Ø													J2Ø											J5Ø			J2Ø
	J2Ø				J6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J7Ø			J2Ø														J2Ø
	J2Ø													J2Ø														J2Ø
	J2Ø													J5Ø			J6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J7Ø			J2Ø
	J2Ø							J4Ø																				J2Ø
1Ø	J2Ø							J2Ø																				J2Ø
	J2Ø			J6Ø	J7Ø			J2Ø												J4Ø				J4Ø				J2Ø
	J2Ø							J2Ø			J6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J7Ø			J2Ø				J2Ø				J2Ø
	J2Ø							J2Ø												J5Ø				J5Ø				J2Ø
	J2Ø							J5Ø																				J2Ø
15	J2Ø											I6Ø	J3Ø	J3Ø	I7Ø													J2Ø
	J2Ø											J2Ø			J2Ø			J6Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J7Ø			J2Ø
	J2Ø			J6Ø	J3Ø	J3Ø	J7Ø					J2Ø			J2Ø													J2Ø
	J2Ø											J2Ø			J2Ø													J2Ø
	J2Ø											J2Ø			J2Ø													J2Ø
2Ø	JØØ	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J1Ø			JØØ	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J3Ø	J1Ø

SAVEは，プログラムのあとにしよう。でも，電源が抜けたりすると大変だから，とりあえずSAVEしちゃおう。SAVEのしかたは，もう知ってるよね。

「ESC」キーを押したあと「STOP」キーを押してプログラムモードへ切りかえよう。

今回は，「DEF　MOVE」文を使うけど，ペンペンもファイターフライもそれぞれ，上下左右4方向，合わせて8つも使う。これを普通に定義していくとメモリーをたくさん使ってしまう。そこで前に勉強した，サブルーチンを使おう。

```
5ØØØ DEF MOVE(N)=SPRITE(N,V
     ,1,4):MOVE N:RETURN
```

つまり，ペンペンとファイターフライがそれぞれ「DEF MOVE」のキャラクターナンバー2，4であるのを利用して，「DEF MOVE」のØから7のうち2と4だけを使う。「N」に2か4を入れて，「V」に方向を入れて「GOSUB　5ØØØ」とすれば，ペンペンか，ファイターフライが「V」の方向に動く，というサブルーチンだ。

```
100 SPRITE ON:CLS:VIEW
110 SC=0:SE=406
```

これはゲーン自体の初期設定だから，一番頭に置く。「VIEW」はBG画面をキャラクター画面に転送する命令だったね。SEは空白の場所の数だ。

```
200 FOR I=1 TO 27:FOR J=1 T
    0 19:IF SCR$(I,J)=CHR$(
    215) THEN LOCATE I,J:PR
    INT " ";
```

CHR$ (215) はリンゴだね。ペンペンが歩いていったあとには，このマークを置くことにする。

一面クリアーしたあとは，このリンゴマークを一度消さなきゃいけないね。それが200行だ。「FOR」が2つあるから，

```
210 NEXT:NEXT:M=0
```

次に乱数で新しいカベを発生させている。230行でカベのX座標，Y座標を出す。その位置に何も置かれてなければ，そこへ新しいカベ「CHR$ (221)」をかく。

●リスト1

```
230 I=RND(24)+2:J=RND(15)+4
250 IF SCR$(I,J)<>" " THEN 230
270 LOCATE I,J:PRINT CHR$(221);:SE=SE-1
```

当然，リンゴでうめなければいけない空白は，一つへるわけだから，「SE=SE－1」。そして，ゲームをRUNさせた直後はここを通らないようにしなければいけないので，

```
150 GOTO 300
```

を追加しよう。300行からがゲームの始まりだ。

007−2
どっちの方向なら動けるか

まず，ペンペンとファイターフライのゲームの開始位置を決よう。230行で「J＝RND (15)＋4」としてあるおかげで，新しくふえるカベは，上の4行には絶対ない。そこで，ペンペンは左上すみ，ファイターフライは右上すみからスタートになるんだ。(この面での空白はSEだから，残りの空白を表わす変数SDにSEを入れよう。)

```
300 SD=SE POSITION 4,24,32
310 POSITION 2,216,32
```

さて，まずペンペンを動かそう。当然コントローラーの「STICK」関数を使うね。

```
400 V=STICK(0):N=4
```

ところが問題は，「STICK」の上下左右と「DEF　MOVE」の上下左右の数値が違うんだ。そこで，

●リスト2

```
410 IF V=8 THEN V=1:GOTO 500
420 IF V=1 THEN V=3:GOTO 500
430 IF V=4 THEN V=5:GOTO 500
440 IF V=2 THEN V=7:GOTO 500
450 V=0:GOTO 510
```

ところが，これだけではまだ移動させてはいけないよ。移動予定先にカベがあるかどうかを調べなきゃいけないからだ。これは，ファイターフライも同じ条件だから，一つのサブルーチンで処理しよう。

```
500 N=4:GOSUB 6000
```

```
51Ø IF F=Ø OR V=Ø THEN
GOSUB 5ØØØ
```

6000行で行けるか行けないかのマーク(フラグといったね。)を0

●リスト3

```
600 X=XPOS(4)/8-2
610 Y=YPOS(4)/8-3
620 FOR I=X TO X+1:FOR J=Y TO Y+1
630 IF SCR$(I,J)=" " THEN LOCATE I,J:PRINT CHR$(21
5);:SD=SD-1:SC=SC+1
640 NEXT:NEXT
650 IF SD<1 THEN 2000
```

にセット。X, Yはスプライト用の座標(Ø～255)をキャラクター用の座標(Ø～27, Ø～23)に直したものだ。

6050行から6080行は,行こうとする先にカベがあるかどうかを調べているんだ。

ペンペンに限っては,新しくできたカベ「CHR$ (221)」は通れなかったから,6070行で調べている。

さて,サブルーチンから帰って,その方向に行けるなら(F=Ø),「MOVE」のサブルーチンへ行ってみて,だめなら(F=1),そのまま次へ行く。

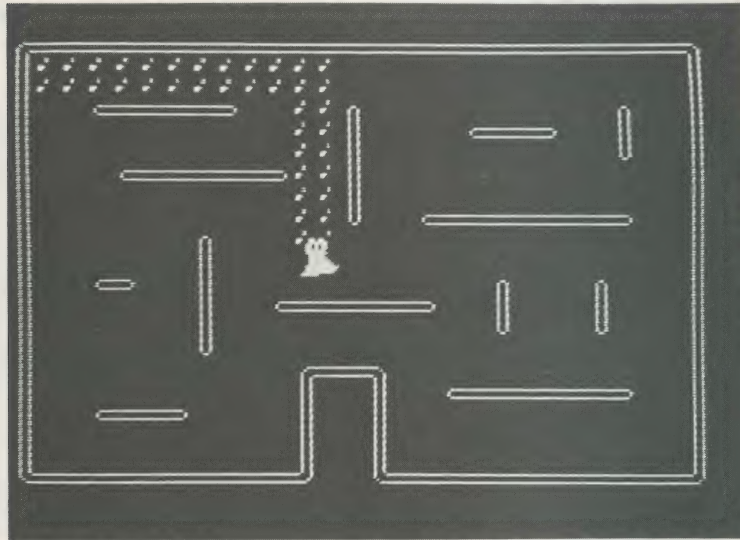
●リスト4

```
700 N=2:V=3+SGN(YPOS(4)-YPOS(2))*2:IF V<>3 THEN GO
SUB 6000:IF F=0 THEN 750
710 V=5+SGN(XPOS(2)-XPOS(4))*2:IF V<>5 THEN GOSUB
6000:IF F=0 THEN 750
720 GOTO 760
750 IF RND(100)<80 THEN 800
760 V=RND(4)*2+1
770 GOSUB 6000:IF F=1 THEN V=0
800 GOSUB 5000
```

この600～640行は6050～6080と同じような構文になってるね。ペ

ンペンのいるキャラクターサイズ2×2の下に，リンゴがなければリンゴを書いて，その分SDをへらしているわけだ。そして，SDが0になれば，全面をリンゴでうめたことになるから，1面終わりの2000行へ飛ぶんだ。この650行を，

```
650 GOTO 400
```

とすると，とりあえずペンペンが動き回り，リンゴがふえていくのが見えるよね。

007−3
敵はおりこうさん

今度はファイターフライを動かそう。

まず，考え方としては，乱数で1，3，5，7を出すというのがある。しかしそれでは，あまりにもファイターフライがバカすぎる。やっぱり，ファイターフライにはもう少し，りこうになってほしい。

そこで，

●リスト5

```
85Ø IF XPOS(2)<XPOS(4)+11 AND XPOS(2)>XPOS(4)-11 A
ND YPOS(2)<YPOS(4)+11 AND YPOS(2)>YPOS(4)-11 THEN
3ØØØ
```

700行と710行でペンペンのいる方向が，どっちかを決めている。このプログラムではY座標方向を優先しているが，700行と710行を入れかえれば，X方向が先になるのはわかるね。

ただ，ファイターフライには，少しおバカさんになってもらうことにする。750行で乱数によって，わざとでたらめな動きをしてもらうのだ。750行の，

IF RND (1ØØ)<8Ø

の「8Ø」を小さくすればするほど，乱数でしか動かない，おバカさんのファイターフライになってしまう。といっても，これを「1ØØ」にすると完全なおりこうさんになるかというと，そうでもない。

ペンペンとファイターフライがスタート位置にいるとして，間にカベがあったらどうだろう。ファイターフライはペンペンに近よってくるが，カベの所までくると，もう動けなくなる。この程度のプログラムでは，カベを回りこんでくる，というのは，なかなかむずかしいからなんだ。だから少しは，乱数で動く可能性を残した方がいいんじゃないかな。

700行と710行ではメモリーの節約のため，6000行と6010行で使ったと同じ「SGN」関数をうまく利用しているね。

これで，ペンペンとファイターフライが自由に動き回れるようになった。いよいよ，ペンペンがファイターフライにつかまったかどうかを調べなきゃいけない。

5000行で決めているように，ペンペンもファイターフライも一回の「MOVE」の命令で8ドット動くんだから，XPOS（4），YPOS（4）のそれぞれ前後±11ドット以内に，ファイターフライが来たときを，つかまったとみなそう。

つまり，GAME　OVERの処理は3000行で行なうわけだ。

●リスト6

```
6000 F=0:X=XPOS(N)/8-2:IF V=3 OR V=7 THEN X=X+SGN(
5-V)
6010 Y=YPOS(N)/8-3:IF V=1 OR V=5 THEN Y=Y+SGN(V-3)

6050 FOR I=X TO X+1:FOR J=Y TO Y+1
6060 IF SCR$(I,J)>CHR$(221) AND SCR$(I,J)<CHR$(232
) THEN F=1
6070 IF N=4 AND SCR$(I,J)=CHR$(221) THEN F=1
6080 NEXT:NEXT:RETURN
```

007−4
最後の仕上げはこうやって

さて，得点はどうしよう？リンゴ1つを書くごとに，プラス1点としよう。とすると当然，リンゴを書いていた630行の一番最後に，

```
SC=SC+1
```

としなけりゃだめだ。その得点を表示するために，

```
95Ø GOSUB 7ØØØ
```

としよう。(7000行がスコア表示のサブルーチンだ。)

また，全部の空白をリンゴでうめると，カベを1個ふやしてはじめからやるわけだ。しかし，スコア (SC) をØに戻してはまずいから，

```
2ØØØ GOSUB 7ØØØ
21ØØ GOTO 2ØØ
```

としなきゃあいけない。また得点を出して，得点表示のサブルーチン7000行と，一面クリアーのお祝いの言葉2050行，次の面をはじめる時に，その言葉を消すための，280行を追加し，3000行からの「END」処理をつけたすと，右のようになる。

このプログラムでは，まだまだメモリーにあまりがあるね。ジャンプ編で勉強する言葉(PLAY) をつけたり，ファイターフライを二匹にしたりと，改造する余地はいっぱいあるんだ。ぜひとも，ガンバロー！

さて，いよいよジャンプ編では，音楽や色に挑戦してみようじゃないか。ゲームも，Tiny ROLL PLAYING GAMEだ！

●リスト7

```
100 SPRITE ON:CLS:VIEW
110 SC=0:SE=406
150 GOTO 300
200 FOR I=1 TO 27:FOR J=1 TO 19:IF SCR$(I,J)=CHR$(
215) THEN LOCATE I,J:PRINT " ";
210 NEXT:NEXT
230 I=RND(24)+2:J=RND(15)+4
250 IF SCR$(I,J)<>" " THEN 230
270 LOCATE I,J:PRINT CHR$(221);:SE=SE-1
280 LOCATE 10,23:PRINT "          ";
300 SD=SE:POSITION 4,24,32
310 POSITION 2,216,32
400 V=STICK(0):N=4
410 IF V=8 THEN V=1:GOTO 500
420 IF V=1 THEN V=3:GOTO 500
430 IF V=4 THEN V=5:GOTO 500
440 IF V=2 THEN V=7:GOTO 500
450 V=0:GOTO 510
500 GOSUB 6000
510 IF F=0 OR V=0 THEN GOSUB 5000
600 X=XPOS(4)/8-2
610 Y=YPOS(4)/8-3
620 FOR I=X TO X+1:FOR J=Y TO Y+1
630 IF SCR$(I,J)=" " THEN LOCATE I,J:PRINT CHR$(21
5);:SD=SD-1:SC=SC+1
640 NEXT:NEXT
650 IF SD<1 THEN 2000
700 N=2:V=3+SGN(YPOS(4)-YPOS(2))*2:IF V<>3 THEN GO
SUB 6000:IF F=0 THEN 750
710 V=5+SGN(XPOS(2)-XPOS(4))*2:IF V<>5 THEN GOSUB
6000:IF F=0 THEN 750
```

●リスト8

```
720 GOTO 760
750 IF RND(100)<80 THEN 800
760 V=RND(4)*2+1
770 GOSUB 6000:IF F=1 THEN V=0
800 GOSUB 5000
850 IF XPOS(2)<XPOS(4)+11 AND XPOS(2)>XPOS(4)-11 A
ND YPOS(2)<YPOS(4)+11 AND YPOS(2)>YPOS(4)-11 THEN
3000
950 GOSUB 7000
999 GOTO 400
2000 GOSUB 7000
2050 LOCATE 10,23:PRINT "GOOD !";
2100 GOTO 200
3000 BEEP:PAUSE 230:BEEP:CLS:SPRITE OFF
3010 LOCATE 7,10:PRINT "SCORE =";SC
3020 LOCATE 5,13:PRINT "HIT ANY KEY !"
3030 PAUSE
3040 RUN
3100 END
5000 DEF MOVE(N)=SPRITE(N,V,1,4):MOVE N:RETURN
6000 F=0:X=XPOS(N)/8-2:IF V=3 OR V=7 THEN X=X+SGN(
5-V)
6010 Y=YPOS(N)/8-3:IF V=1 OR V=5 THEN Y=Y+SGN(V-3)

6050 FOR I=X TO X+1:FOR J=Y TO Y+1
6060 IF SCR$(I,J)>CHR$(221) AND SCR$(I,J)<CHR$(232
) THEN F=1
6070 IF N=4 AND SCR$(I,J)=CHR$(221) THEN F=1
6080 NEXT:NEXT:RETURN
7000 LOCATE 8,21:PRINT "SCORE =";SC;:RETURN
```

作戦資料 101 緊急情報

わが地球防衛軍スタッフの連日にわたるファミコンとの闘いで，以下の事が判明した。

CALL

① Ver. 2でも後期に印刷されたマニュアルには書かれているんだが，「CALL」という命令がかくされていた。これはマシン語プログラムを動かす命令である。使い方は，ジャンプ編でくわしくのべよう。

②他にも「SPC」と「TAB」という命令が登録されていた。ただし，これは名前が登録されているだけで使えない。Ver. 4以降にでも使うつもりかもしれない。

③　バグ情報ー1

TAPEばけ

Ver. 3で作ったプログラムをテープへSAVEしたものをVer. 2でLOADすると，プログラムがメチャメチャになってしまう，というのは，ホップ編や，ゲームプログラム集001でもすでに説明したが，やっと原因がわかったので，説明しよう。

コンピューターが数字をメモリーへ入れる時，たとえば

```
LOCATE 0,1
```

とかいうものの「0」や「1」は，まず最初の場所に，「次は10進法の数字だよ」というしるし（コード）を書いて，そのあと2バイトを使ってその数字を入れている。(つまり1つの数字を記憶するのに3バイト必要だ）ところが，Ver. 3では，何を考えたか，1ケタの数字を記憶する時に限って，1バイトしか使わないように変更してしまったんだ。たしかに，メモリー数の少ないファミコンでは，メモリーの節約は非常にありがたいんだけど，この方式で書かれたテープを，Ver. 2で読むと，大変なことになる。Ver. 2では，数字をあらわすのに必ず3バイト使っているから，プログラムを見に行く順番が狂ってしまうのだ。

[命令] [す][う][じ] [:] [命令]　Ver. 2

[命令] [1ケタの数字] [:] [命令]　Ver. 3

Ver. 3をもっている人はためしてほしい。

```
10 LOCATE 10,12
```

```
10 LOCATE 0,2
```

この2つのプログラムで残りバイト数(PRINT　FREでよかったね)が変る事がわかるだろう。

対策としては

```
10 LOCATE 0,1
```

のような1ケタの数字の前に，必ず

```
10 LOCATE &H0,&H1
```

のように「&H」という文字を入れておけば大丈夫だ。この「&H」というのは16進数という数のかぞえ方を意味するやっかいなものだ。(ジャンプ編でくわしく説明する)

ただし，どんなことがあっても，2ケタ以上の数字の前にはつけない事(10→&H10のように)，数字の中身が変ってしまうからだ。

```
10 LOCATE 10,12
10 LOCATE &H0,&H2
```

この二つでは残りバイトが一緒だというのは調べればすぐわかる。

COLOR文の位置ずれ

④ バグ情報—2

以下のプログラムを実行してみよう。

```
10 CLS
20 LOCATE 8,8
30 PRINT "TEST1"
40 LOCATE 8,9
50 COLOR 8,9,3
60 PRINT "TEST2"
```

「LOCATE」文で，X座標を「8」に指定しているにもかかわらず，60行のPRINT文の文字列は，ずれて表示されている。つまり，COLOR命令を実行するとカーソルの位置がずれてしまうのだ。

```
10 LOCATE 8,8
20 A=POS(0)
30 COLOR 8,8,3
40 B=POS(0)
50 CLS
60 PRINT A,B
```

このプログラムでわかるだろう。このバグはVer.3では解消され

ているが，Ver. 2ではLOCATE命令をともなったCOLOR文は必ず，PRINT文のうしろに置くことが必要だ。

カタカナの化け

⑤　バグ情報—3

```
1Ø READ A$
2Ø PRINT A$
3Ø DATA ャ
```

として「RUN」してみよう。ちゃんと「ャ」と表示されるはずだ。ここで「LIST」をとってみると，

```
1Ø READ A$
2Ø PRINT A$
3Ø DATA READ
```

となっている。

これは，「LIST」の処理ルーチンにバグがあるため，「DATA」のうしろにカタカナがあると，ものによっては中間言語（ファミリーベーシック・ジャンプ編を見て！）と感違いしてしまうせいだ。

```
3Ø DATA "ャ"
```

のようにすれば問題はないが，普通ほとんどのパソコンも，DATA文の文字列は「〃〃」で囲まれていなくてもよいという約束なので困ってしまう。「ャ」以外にもこういう変化をするカタカナはいくつもある。探してみよう。また，変化する時としない時がある。なお，このバグはVer. 3でもとれていない。

カナ文字モードの RETURN KEY

⑥　バグ情報—4

どのコンピューターも，「英数字モード」と「カナ文字モード」のどちらの状態であっても，RETURNキーの機能は同じのはずだが，我々スタッフがEDIT（行の中身を修正する）中に，「カナ文字モードの状態でRETURNキーを押したあと「LIST」を取ると，修正されていない（RETURNキーの機能が働いてない）という現象が何回かあった。あとでのべる「熱暴走」のせいかもしれないが，重要な修正は「英数字モード」にしたあとRETURNキーを押した方が安心だ。

⑦　バグ情報—5

次のプログラムを入力してほしい。

```
1ØØ READ A$,B
```

```
110 PRINT A$,B
200 DATA "C"
210 DATA 1
```

DATA
READ

こうすると「A$」にはちゃんと「C」という文字が入っているけれど，数値変数「B」にはDATA文にある「1」ではなくて「0」が入っている。

ところが，この200行と210行を

```
200 DATA "C",1
```

または，

```
200 DATA C
210 DATA 1
```

のようにすると，「B」にはちゃんと「1」が入っている。

これは，おそらくDATA文の中で「" "」のあとに何もない時は，「，」の役割りも一緒にしているせいだと思われる。

つまり，

```
200 DATA "C",
```

と書いたのと同じわけだ。

コンピューターは「C」という文字を読んだ（READ　A$）あと，その「C」のうしろに「，」があると思う。ところが「，」のうしろには何もないので，とりあえず「0」を次のREADに代入してしまうというわけだ。

おもしろいのは「" "」のうしろに「，」があって，他のデータが続くときは「" "」は「，」の働きをしなくなっている。

このバグを防ぐには，

```
200 DATA "C",1
```

のように「" "」のうしろに他のデータをつけるか，「" "」を使わないか，しかないが，バグ情報―3に書いたように，「" "」をつけないとカタカナの文字列がLISTをとるとバケてしまう。DATAをREADする時，文字変数よりあとに数値変数を置くのも，プログラムの内容によっては不可能な時もある（RESTOREなどの時）。

「ソフト集002」では，ダミーのデータを一個おいているが，困ったバグだ。

このバグは，Ver.3ではなくなっているようだ。

⑧　バグ情報－6

これは，「バグ」ではないが，それ以上に

注意！

要　注　意　！

「熱暴走」という言葉を知っているだろうか。古いパソコンではよくあった。長時間コンピューターの電源を入れているうちに，コンピューターが熱を持ち，その熱がコンピューターに影響を与え，正常な動作をしなくなることがある。

現在，わがスタッフは12台のファミコンを持ち，日夜開発とデバッグにはげんでいるのだが，その内の約半数が時々暴走してしまうのだ。

プログラムを入力中に，突然，あの恐怖のオープニング画面と共に「ピコピコ」と音が出てきたり，カーソルが消えてキーボードをたたいても何の反応もなくなってしまうのだ。

たしかに，徹夜を続けて何日間も電源を入れっぱなしの，かなりハードな使い方をしているのは事実だが，これではあまりにもスタッフがかわいそう。

電源はこまめに切りましょう。また，コンセントにさし込んだ電源アダプターも，ファミコンを使わない時はぬいておきましょう。

それにしても，少々故障率が高すぎる気もするのだが……。

作戦資料　102　ROMとRAM

プログラムを作ったあと，テープにSAVEしないと，電源を切ったら全部消えちゃうのは，もうわかったよね。じゃあ，ゲームのカセットや，ファミリーベーシックのカセットは，なぜ電源を入れるたびに，同じ働きをするんだろう？ゲームカセットの中にもゲームプログラムが入っているはずだし？

そう考えた君は，するどい！　相当にするどい！

パソコンのMSXのゲームも，テープで発売されているのもあれば，カセットで発売されているのもある。どこがどう違うんだろう？

君の考えた通り，カセットのゲームも，テープのゲームも，本当

はベーシック自体も，一つのプログラムなんだ。じゃあ，なぜ消えるプログラムと，消えないプログラムがあるんだろう？

それは，プログラムのせいではなく，プログラムを書きこんでいる「もの」が違うせいなんだ。

例えば，クーラーなんかにも，マイコン内蔵型というのが発売されてるよね。まさか，クーラーに入ったマイコンを使ってゲームをする人は，いないだろ。「ドンキー・コング」のカセットで「ゼビウス」はできないだろう。印刷された「本」と，まっ白な「ノート」の違いは，わかるだろう。１冊の本の中身は，いつまでたっても変わらない。それにくらべて，ノートにはいろんな物をかいたり消したりして，別の物を書ける。

本とノート

つまり，本やクーラーのマイコンのように，使い道が完全に決まっているものは，最初から印刷した方がいいだろう。白紙の本を買ってきて，それにテープか何かから，内容を写したあとで読む，なんて，だれもやんないよね。

コンピューターのプログラムを記憶しておく「容器」には「ROM」と「RAM」という二種類あるんだ。

Read　Only　Memory
読みだし専用メモリー

Randam　Access　Memory
自由に使えるメモリー

のそれぞれ頭文字をとった略称なんだ。

「ROM」にプログラムを入れる（焼き込む，ともいう）には，特別な機械（といっても，最近は安いもので４～５万円であるけど）が必要だ。そのかわり，一回焼き込むと電源を切っても中身は保存されている。

普通のコンピューターでは「ROM」からプログラムを読み出すことはできても，書き込むことはできない。つまり，読み出し専用メモリー，というわけだ。

「RAM」の方は，コンピューターで自由に書いたり読んだりできるけど，電源が切れると，中身も消えちゃう。電源を切ると，全ページに消しゴムを使うのと同じだから，電源を入れたときは，いつも白紙のノートとして使えるんだ。

だから，「RAM」は，コンピューターのプログラムのように，そのつどに別の仕事をさせるために，中身がいつも変わってないといけないものを記憶するときに使うんだ。

ゲームカセットやベーシックは，この「ROM」の中身にプログラムを入れていたので，電源を切っても中身が消えなかったわけなんだ。

ただ，ゲームカセットでは「ROM」だけしか使わないかというと，そうでもないんだ。得点や，主人公が今どこにいるか，というように刻々と変化するデーターは，いつも，書き込み，読み出しをしなければいけないだろう。

ファミコンのRAM

ファミコン内部には，そういうデーターを入れる「RAM」（ワークエリア）が，ちゃんと入っている。

せっかく出した「ハイスコア」が，電源を切るとなくなっちゃうのは，やはり，得点を書きこんでいるのが「RAM」だから，というわけだ。

また，「ROM」に一回書き込むと，絶対中身が消えないか，というとそうでもないんだ。「ROM」の材料によっては，家庭用の螢光灯に長い時間くっつけておくと消えてしまうものや，特殊な機械で，3～4回は書きかえることができるものもある。

でも，お願いだから，実験はしちゃだめだよ！もう一台，買わなきゃならないよ……。

作戦資料　103　コントロールキーの使い方

Ctr-

ホップ編でも書いたけど，このコントロールキー，なくても別にかまわない。でも，作戦005で使った「Ctr—D」なんて，すごく便利だ。

そこで，ここでは，コントロールキーの使い方をくわしく書いてみよう。

「Ctr—D」というのは，「CTR」キーを押しながら，「D」のキーを押すことだったね。

この「Ctr—」で使えるキーは次の14個だ。

A，C，D，E，G，H，J，K，L，M，R，V，W，Z

普通のパソコンだと「カナ文字」モードだと使えないことが多いけど，ファミコンでは大丈夫なんだ。

まず，キーボード上の特殊キー（ホップ編参照）と同じ働きをするのが8個だ。

Ctr—H　「DEL」キー
Ctr—J　カーソルキーの「▼」
Ctr—K　「HOME」キー
Ctr—L　「SHIFT」キーを押しながら「CLR・HOME」キー
Ctr—M　「RETURN」キー
Ctr—R　「INS」キー
Ctr—V　カナ文字モードにする
Ctr—W　英数字モードにする

このうち「Ctr—V」と「Ctr—W」は「カナ」キーと同じだけど，「カナ」キーは，一回押すごとに，カナ文字モードと英数字モードが切りかわったのに対し，それぞれの役割を分けているんだ。

さて，何かのプログラムを打ち込んでみよう。どこか文字の書かれている所で，

Ctr—E

を押してみよう。その行のカーソルからあとが，全部消えたのがわかるかな。

今度は別の文字の上へカーソルを持っていって，

Ctr—Z

と押してみる。これは，その行だけじゃなく，カーソルの下の行も全部一変に消えてしまった。

ただし，この「Ctr—E」，「Ctr—Z」ともに，表示されているのを消しただけで，リターンキーで押さない限り，プログラムは消えてはいないんだ。場所を限定した「CLR」キーだと思えばいいかな。

Ctr—G

とすると，「ビー」という音がしたね。ベーシックの命令の「BEEP」と同じ働きだね。

Ctr—D

これは，作戦005で何回も使ったね。スプライトの所で，すべての

表示を初期設定（電源を入れたときと同じ状態）にする役目だ。

CGEN 2

SPRITE OFF

Ctr—A（INSモード）の解除

カラーパレットをスプライト・キャラクター面とともに，キャラクター用のパレットコード1にする。

の4つの働きを一度にしてしまうのだ。

Ctr—A

「INS」キーは，一回押すと空白を一個，カーソルの手前にわりこます働きを持っていた。ところが，この「Ctr—A」は少し違うんだ。

TEST

と文字を入力して，カーソルを「S」の上に持っていこう。そこで，

Ctr—A

とする。何も変化しない，ようにみえる。

ここで，たとえば「D」のキーを押してみよう。

TEDST

となって，カーソルは「S」の上だね。いろんなキーを押してみよう。次々とその文字が「S」の前にわり込んでくる。

この状態を「INS」モード，つまりこれが「わり込みモード」なんだ。ここでもう一回，

Ctr—A

を押すと，今度はこの「INS」モードが解除されている。「カナ」キーと同じように，一回押すごとに，ON，OFF と変化するわけなんだ。

Ctr—D

でも，この「INS」モードを解除できるよ。

最後にコントロールコード Ctr—C は，プログラム入力中に操作すると，その行は登録されません。（プログラム入力中に STOP キーを押すのと同じ働きです。）

キャラクタテーブルA

これらのキャラクタは、主にアニメキャラクタとして使用します。
各キャラクタの四隅に表記している数字は、DEF SPRITE文のCHR$（n）で使用する10進数です。

検印省略　　NDC 548

ファミリーベーシック ステップ

昭和60年6月24日　発行

定価　800円

著者　地球防衛軍

発行者　小川茂男

発行所　㈱誠文堂新光社
東京都千代田区神田錦町1—5—5
郵便番号 101
電話 東京 (292) 1211
振替口座 東京 7—6294
印刷・広研印刷株式会社
製本・岡嶋製本工業

ISBN4-416-18527-8 C2055

ファミコン大作戦
ファミリーベーシック™ステップ

誠文堂新光社 ISBN4-416-18527-8 C-2055 ¥800E 定価 800 円

ファミコン大作戦
ファミリーベーシック™ステップ
誠文堂新
7